SunFounder PiDog Kit

Release 1.0

sunfounder

Feb 02, 2024

CONTENTS

Thank you for choosing our PiDog.

Note: This document is available in the following languages.

-
-
-

Please click on the respective links to access the document in your preferred language.

PiDog is a Raspberry Pi pet robot with aluminum alloy structure. It can act as a mechanical pet, show cuteness to you, and interact with you.

It is equipped with a camera module, which can perform color recognition, face detection and other projects; 12 metal gear servos support it to walk, stand, sit, shake its head, and pose in various poses; The ultrasonic module on the head enables it to quickly detect obstacles ahead; Special touch sensors allow it to respond to your touch; The Light Board on the chest can emit colorful light effects, and with the speaker equipped with the robot HAT, PiDog can express emotions such as happiness and excitement. In addition, PiDog is also equipped with a sound direction sensor and a 6-DOF IMU module to realize more complex and interesting usage scenarios.

If you have any questions, please send an email to service@sunfounder.com and we will respond as soon as possible.

Content

CHAPTER

ONE

ASSEMBLE VIDEOS

Before assembling the PiDog, please first verify that all parts and components have been included. If there are any missing or damaged components, please contact SunFounder immediately at service@sunfounder.com to resolve the issue as soon as possible.

Please follow the steps on the following PDF for assembly instructions:

- `Component List and Assembly Instructions`.

Mount Raspberry Pi Zero W on PiDog

If your mainboard is a Raspberry Pi Zero W, here are the steps to install it on the PiDog.

Afterward, you can continue following the instructions in the video below from **2:28** onwards to assemble it.

Assembly Tutorial Video(Raspberry Pi 4/3/1 Model)

This video will walk you through the process of assembling your robot from scratch.

In this tutorial, you will learn:

- **Preparation**: We'll introduce you to all the tools and parts needed, ensuring you're fully equipped before starting the assembly.
- **Assembly Steps**: We'll demonstrate each assembly step in a systematic manner.
- **Tips and Considerations**: Throughout the process, we'll share essential tips and tricks to help you avoid common mistakes and ensure your robot operates smoothly.
- **Zeroing a Servo**: Before fixing each servo, it needs to be zeroed first. The steps for zeroing are to first install the Raspberry Pi OS, then install the required modules, and then run a script (set the angle of all PWM pins to 0). After that, plug in the servo wire to zero the servo.

The assembly process for Pidog is quite long, so we have divided it into two videos. The first video covers assembling Pidog's body and four legs.

The second video covers assembling the head and calibration.

CHAPTER

TWO

PLAY WITH PYTHON

If you want to program in python, then you will need to learn some basic Python programming skills and basic knowledge of Raspberry Pi, please configure the Raspberry Pi first according to *1. Quick Guide on Python*.

2.1 1. Quick Guide on Python

This section is to teach you how to install Raspberry Pi OS, configure wifi to Raspberry Pi, remote access to Raspberry Pi to run the corresponding code.

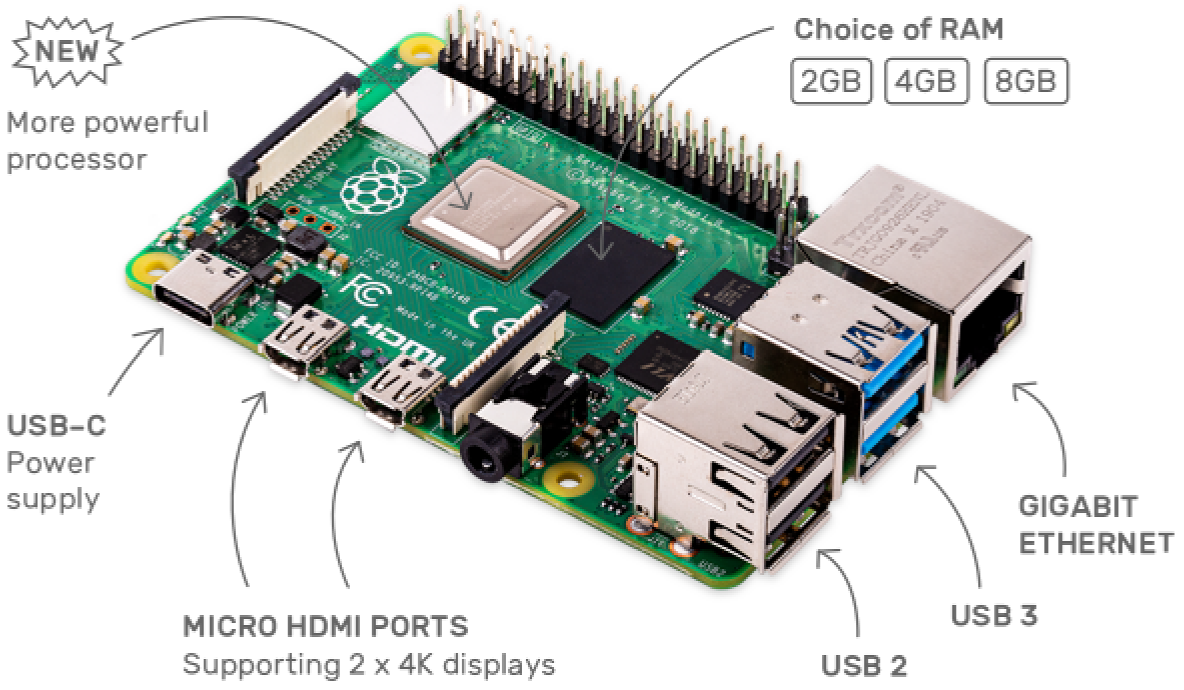
If you are familiar with Raspberry Pi and can open the command line successfully, then you can skip the first 3 parts and then complete the last part.

2.1.1 1. What Do We Need?

Required Components

Raspberry Pi

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python.



Power Adapter

To connect to a power socket, the Raspberry Pi has a micro USB port (the same found on many mobile phones). You will need a power supply which provides at least 2.5 amps.

Micro SD Card

Your Raspberry Pi needs an Micro SD card to store all its files and the Raspberry Pi OS. You will need a micro SD card with a capacity of at least 8 GB

Optional Components

Screen

To view the desktop environment of Raspberry Pi, you need to use the screen that can be a TV screen or a computer monitor. If the screen has built-in speakers, the Pi plays sounds via them.

Mouse & Keyboard

When you use a screen , a USB keyboard and a USB mouse are also needed.

HDMI

The Raspberry Pi has a HDMI output port that is compatible with the HDMI ports of most modern TV and computer monitors. If your screen has only DVI or VGA ports, you will need to use the appropriate conversion line.

Case

You can put the Raspberry Pi in a case; by this means, you can protect your device.

Sound or Earphone

The Raspberry Pi is equipped with an audio port about 3.5 mm that can be used when your screen has no built-in speakers or when there is no screen operation.

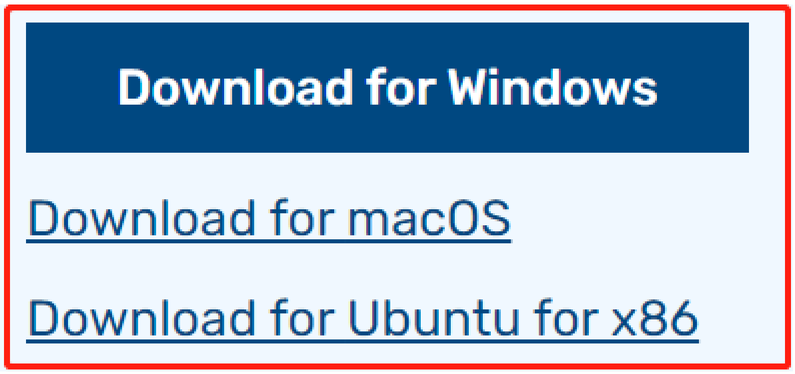
2.1.2 2. Installing the OS

Required Components

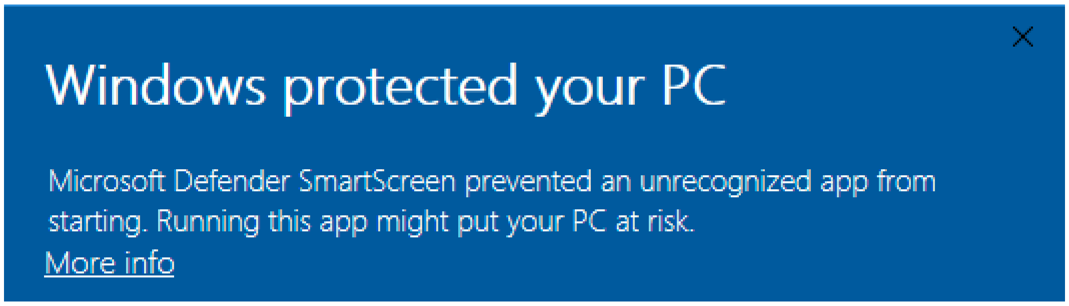
- Raspberry Pi 5B
- A Personal Computer
- A Micro SD card

Installation Steps

1. Visit the Raspberry Pi software download page at Raspberry Pi Imager. Choose the Imager version compatible with your operating system. Download and open the file to initiate installation.



2. A security prompt may appear during installation, depending on your operating system. For example, Windows might display a warning message. In such cases, select **More info** and then **Run anyway**. Follow the on-screen guidance to complete the installation of the Raspberry Pi Imager.

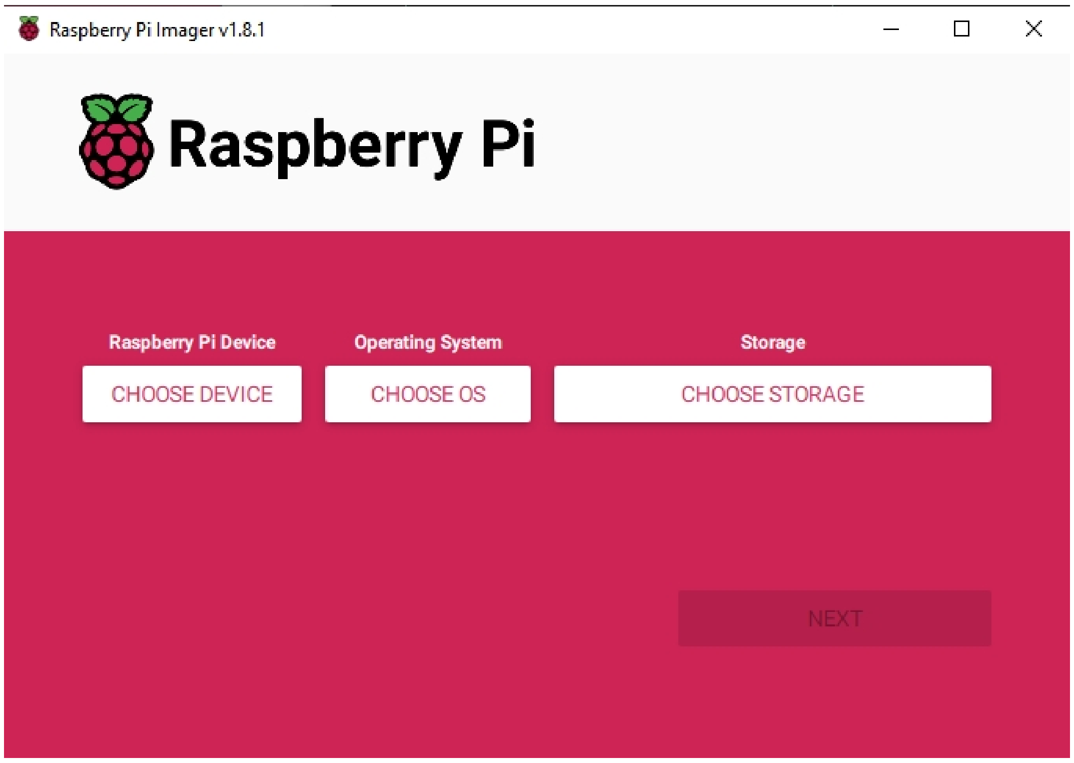


Windows protected your PC

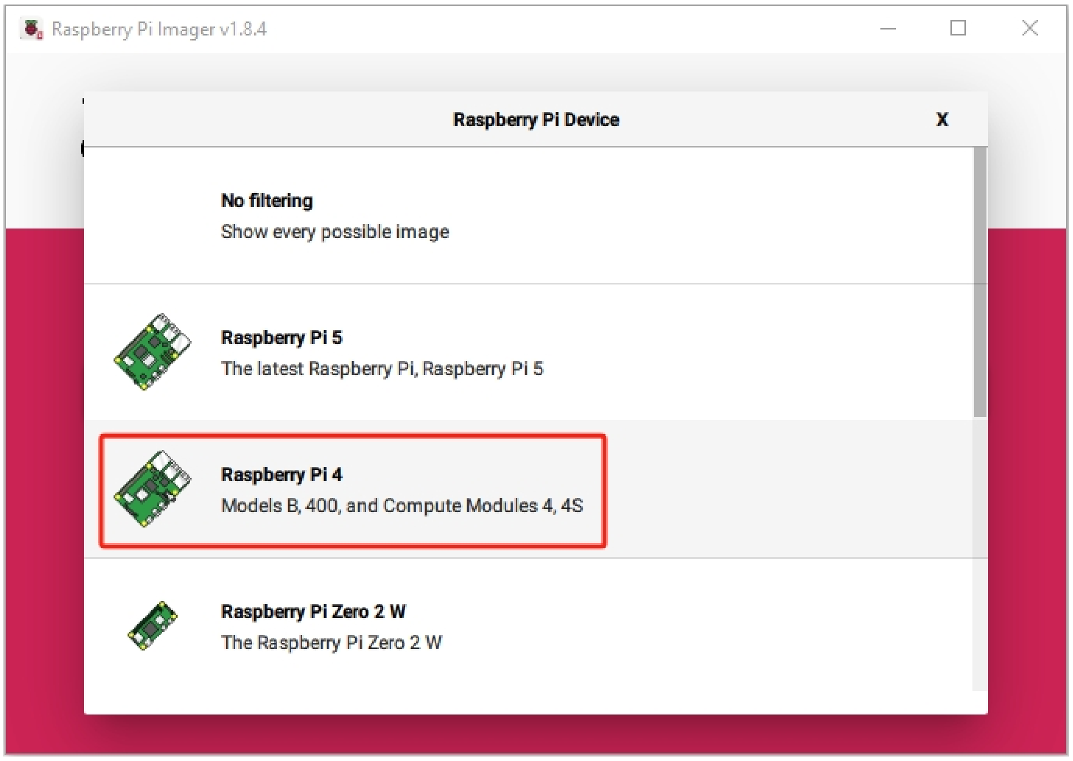
Microsoft Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.

More info

3. Insert your SD card into your computer or laptop's SD card slot.
4. Launch the Raspberry Pi Imager application by clicking its icon or typing `rpi-imager` in your terminal.



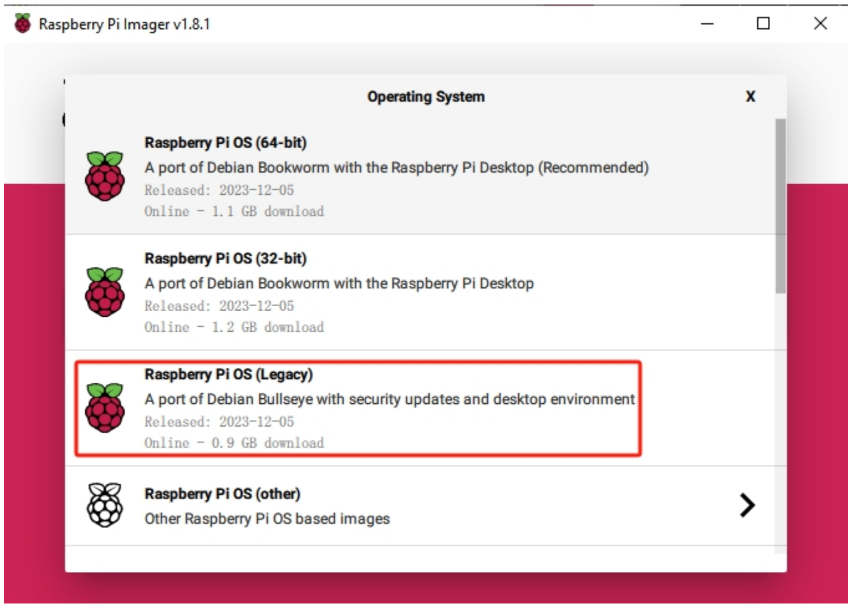
5. Click **CHOOSE DEVICE** and select your specific Raspberry Pi model from the list (Note: Raspberry Pi 5 is not applicable).


Raspberry Pi Imager v1.8.4
Raspberry Pi Device
X
No filtering
Show every possible image
Raspberry Pi 5
The latest Raspberry Pi, Raspberry Pi 5
Raspberry Pi 4
Models B, 400, and Compute Modules 4, 4S
Raspberry Pi Zero 2 W
The Raspberry Pi Zero 2 W

6. Select **CHOOSE OS** and then choose **Raspberry Pi OS (Legacy)**.

Warning:

- Please do not install the **Bookworm** version as the speaker will not work.
- You need to install the **Raspberry Pi OS (Legacy)** version - **Debian Bullseye**.



Raspberry Pi OS (64-bit)
A port of Debian Bookworm with the Raspberry Pi Desktop (Recommended)
Released: 2023-12-05
Online - 1.1 GB download

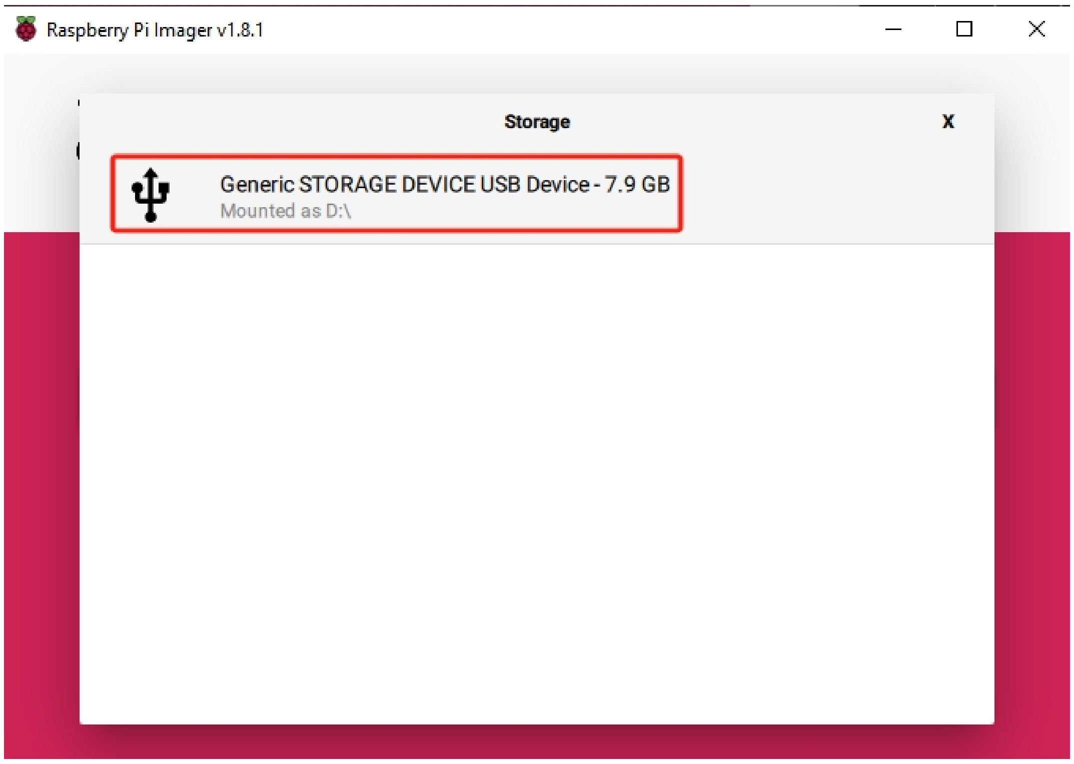
Raspberry Pi OS (32-bit)
A port of Debian Bookworm with the Raspberry Pi Desktop
Released: 2023-12-05
Online - 1.2 GB download

Raspberry Pi OS (Legacy)
A port of Debian Bullseye with security updates and desktop environment
Released: 2023-12-05
Online - 0.9 GB download

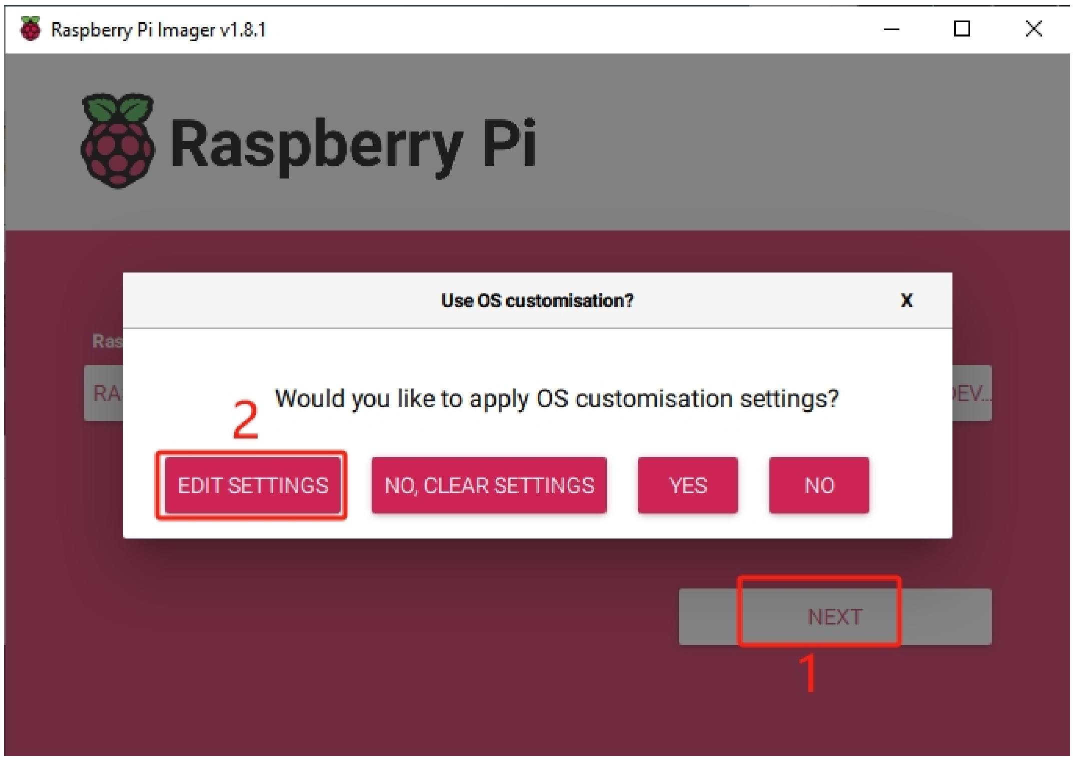
Raspberry Pi OS (other)
Other Raspberry Pi OS based images

7. Click **Choose Storage** and select the appropriate storage device for the installation.

Note: Ensure you select the correct storage device. To avoid confusion, disconnect any additional storage devices if multiple ones are connected.

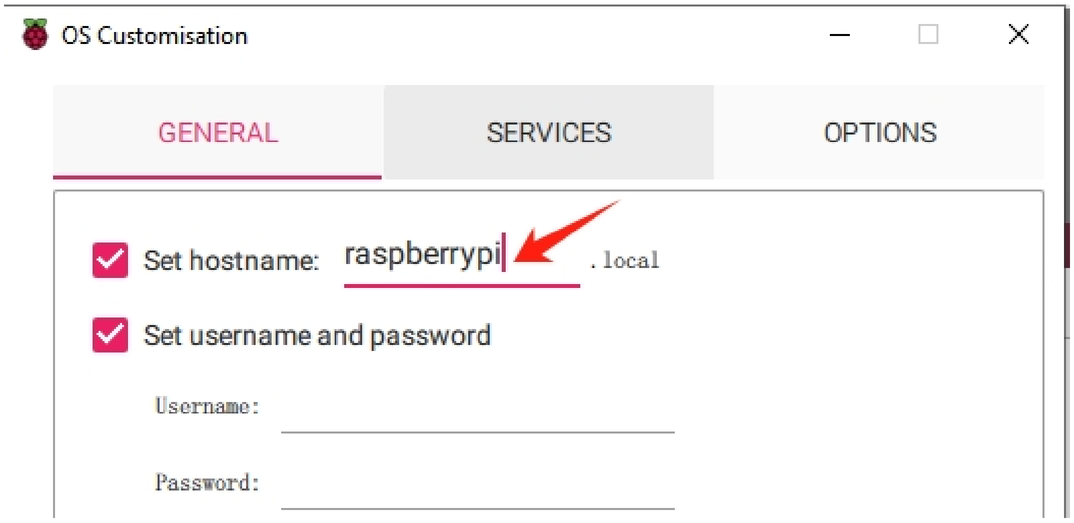

Raspberry Pi Imager v1.8.1
Storage
X
Generic STORAGE DEVICE USB Device - 7.9 GB
Mounted as D:\

8. Click **NEXT** and then **EDIT SETTINGS** to tailor your OS settings. If you have a monitor for your Raspberry Pi, you can skip the next steps and click 'Yes' to begin the installation. Adjust other settings later on the monitor.


Raspberry Pi Imager v1.8.1
Raspberry Pi
Use OS customisation?
X
Would you like to apply OS customisation settings?
2
EDIT SETTINGS
NO, CLEAR SETTINGS
YES
NO
NEXT
1

9. Define a **hostname** for your Raspberry Pi.

Note: The hostname is your Raspberry Pi's network identifier. You can access your Pi using `<hostname>.local` or `<hostname>.lan`.



Set hostname: raspberrypi .local

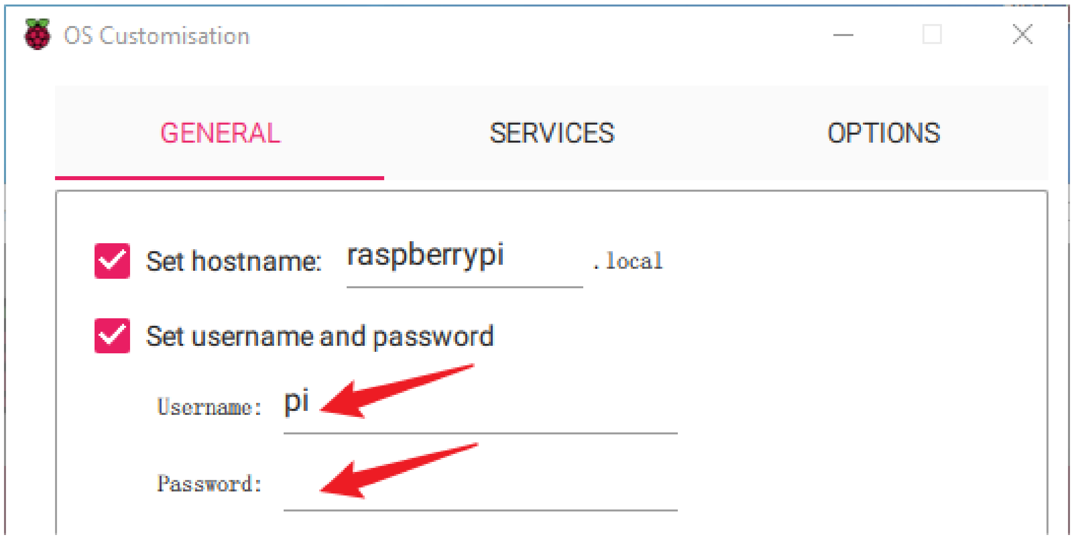
Set username and password

Username:

Password:

10. Create a **Username** and **Password** for the Raspberry Pi's administrator account.

Note: Establishing a unique username and password is vital for securing your Raspberry Pi, which lacks a default password.



Set hostname: raspberrypi .local

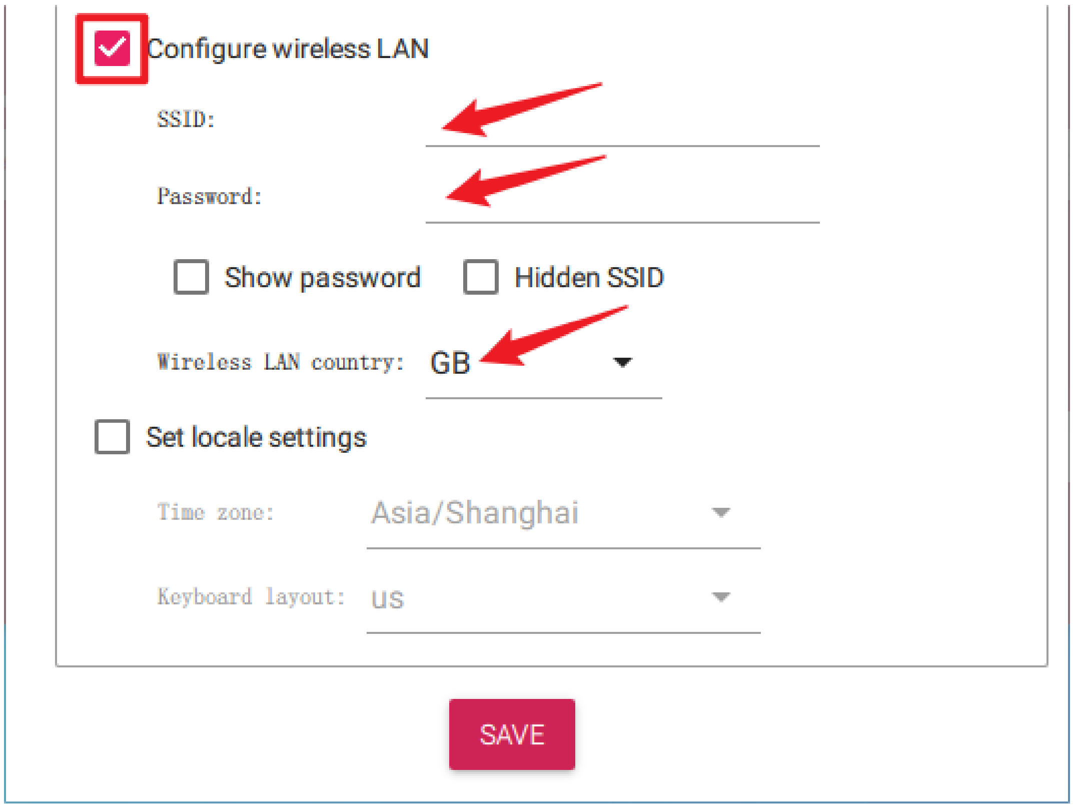
Set username and password

Username: pi

Password:

11. Configure the wireless LAN by providing your network's **SSID** and **Password**.

Note: Set the `Wireless LAN country` to the two-letter ISO/IEC alpha2 code corresponding to your location.


☑ Configure wireless LAN

SSID:

Password:

☐ Show password ☐ Hidden SSID

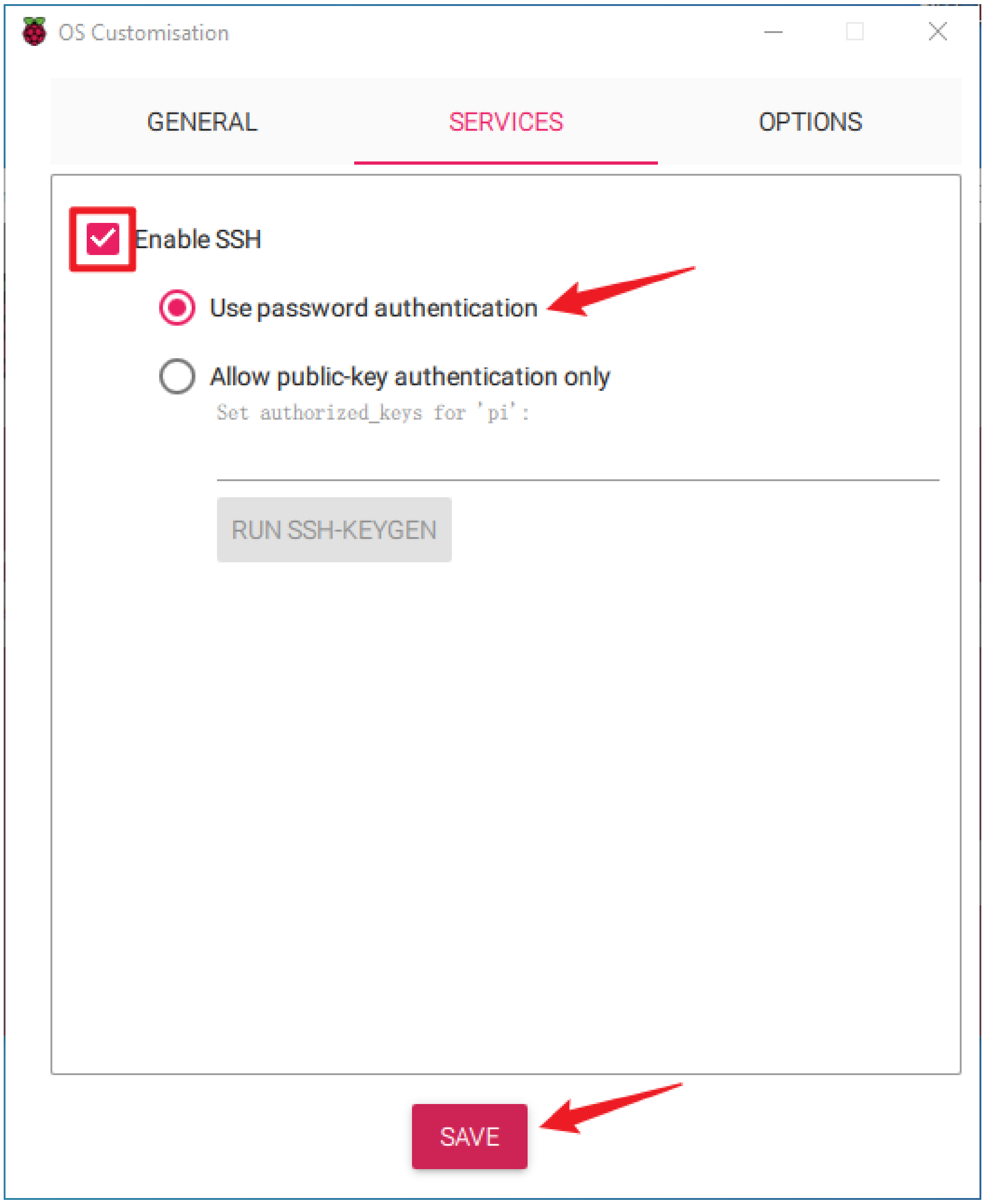
Wireless LAN country: GB

☐ Set locale settings

Time zone: Asia/Shanghai

Keyboard layout: us

12. Click **SERVICES** and activate **SSH** for secure, password-based remote access. Remember to save your settings.



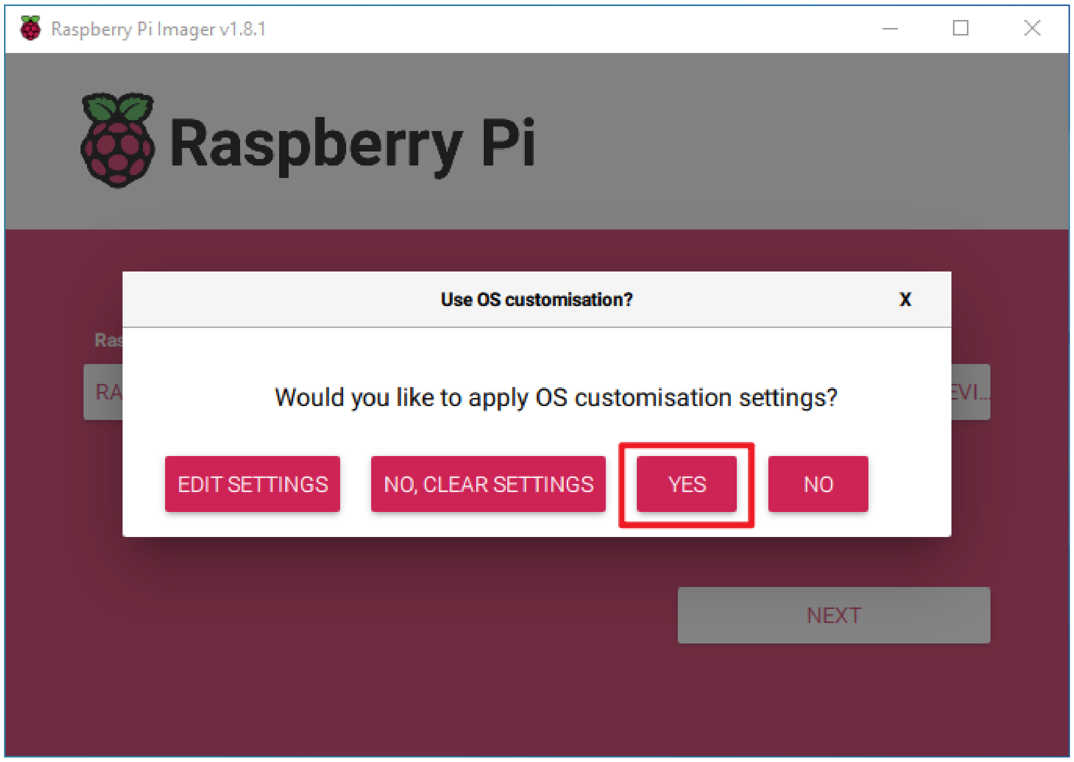
Enable SSH [✓]

[✓ Use password authentication]

Allow public-key authentication only

Set authorized_keys for 'pi':

13. Confirm your selected settings by clicking **Yes**.

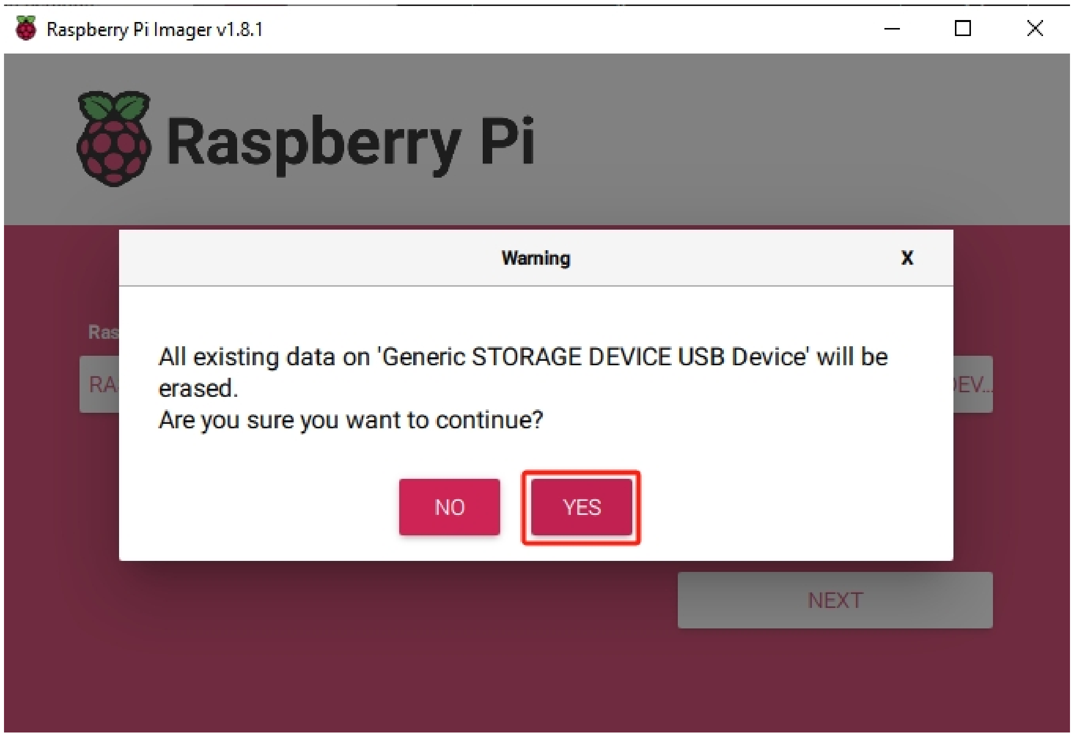


Use OS customisation?

Would you like to apply OS customisation settings?

EDIT SETTINGS | NO, CLEAR SETTINGS | YES | NO

14. If the SD card contains existing data, ensure you back it up to prevent data loss. Proceed by clicking **Yes** if no backup is needed.

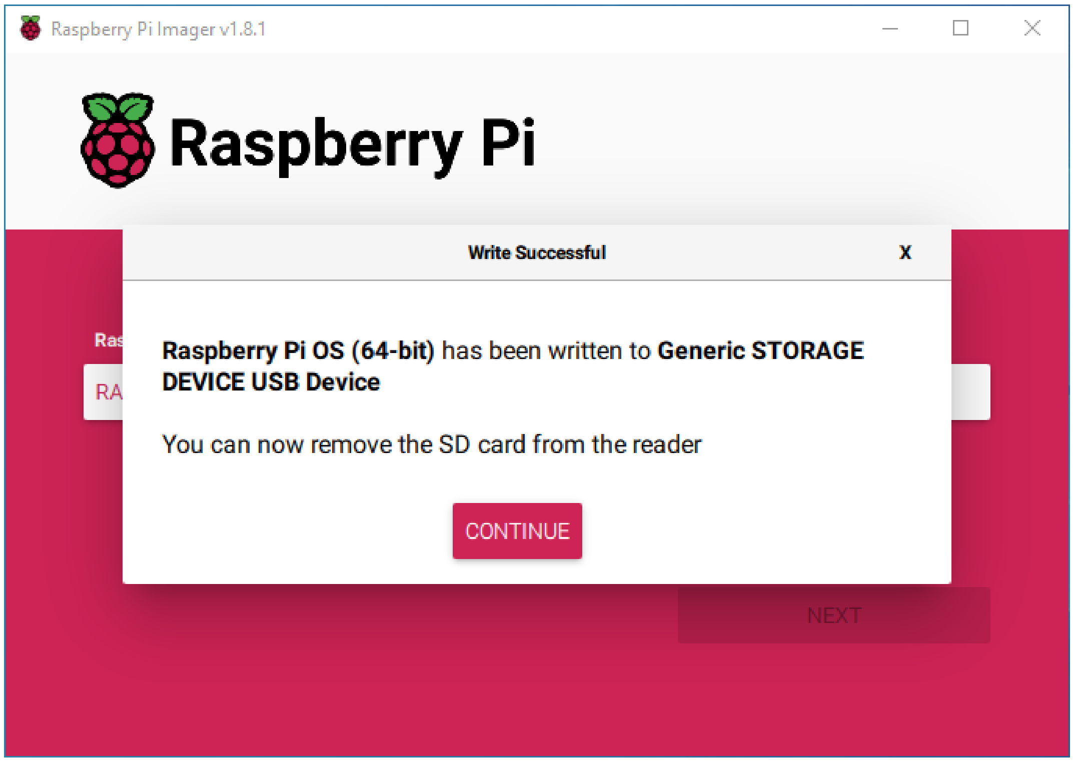


Warning

All existing data on 'Generic STORAGE DEVICE USB Device' will be erased.
Are you sure you want to continue?

NO | YES

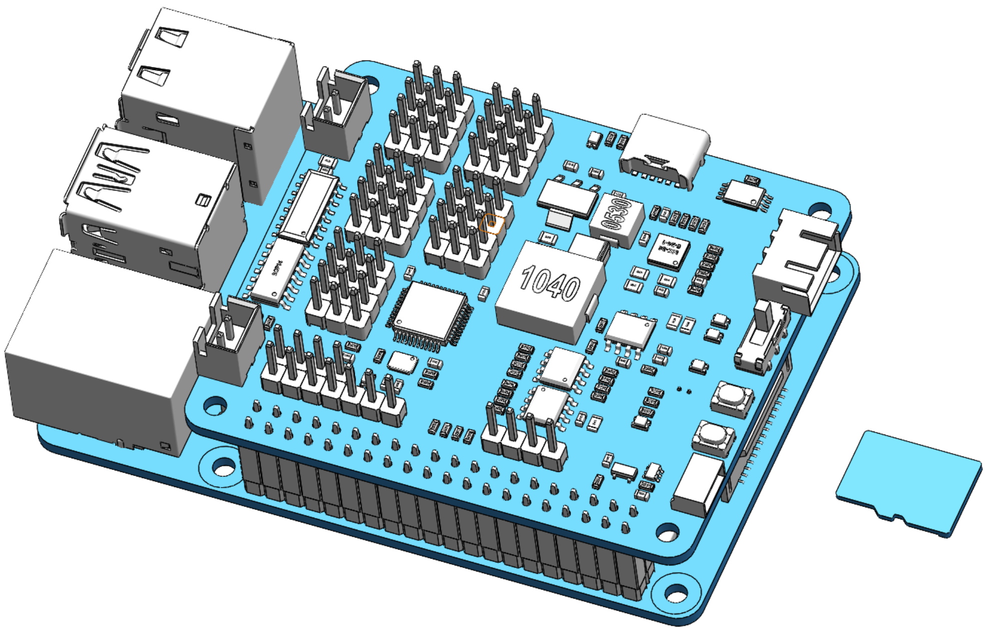
15. The OS installation process will commence on the SD card. A confirmation dialog will appear upon completion.



Raspberry Pi OS (64-bit) has been written to **Generic STORAGE DEVICE USB Device**

You can now remove the SD card from the reader

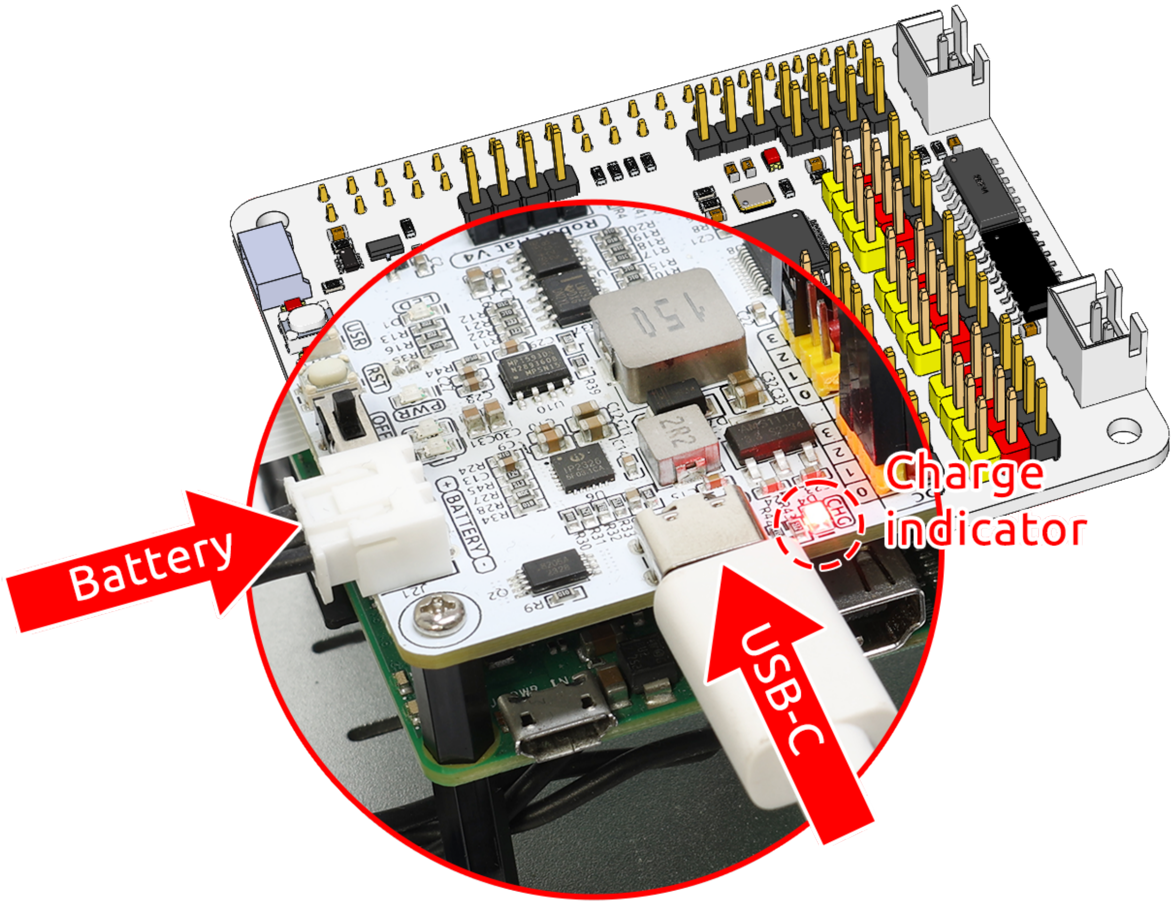
16. Insert the SD card set up with Raspberry Pi OS into the microSD card slot located on the underside of the Raspberry Pi.



2.1.3 3. Power Supply for Raspberry Pi (Important)

Charge

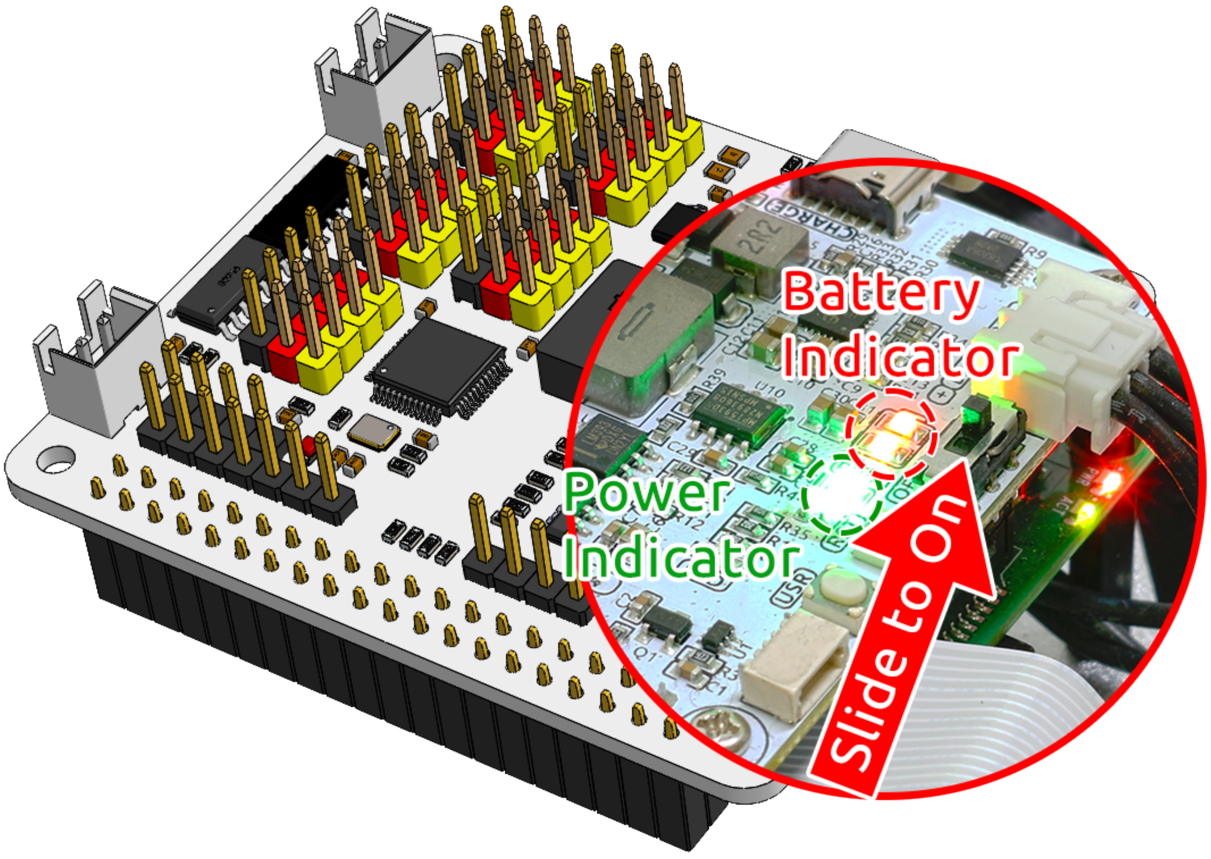
Insert the battery cable. Next, insert the USB-C cable to charge the battery. You will need to provide your own charger; we recommend a 5V 3A charger, or your commonly used smartphone charger will suffice.



Note: Connect an external Type-C power source to the Type-C port on the robot hat; it will immediately start charging the battery, and a red indicator light will illuminate.When the battery is fully charged, the red light will automatically turn off.

Power ON

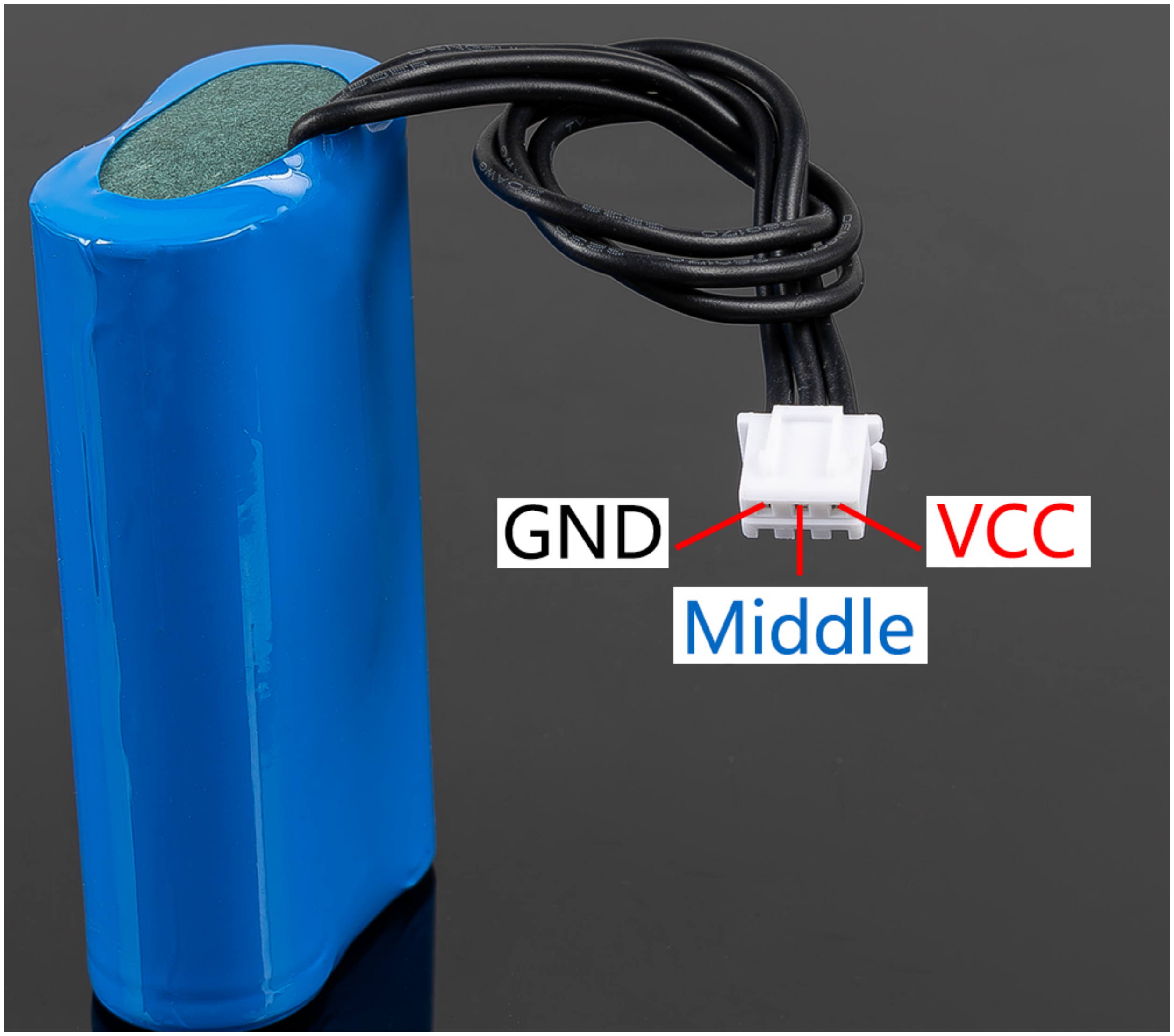
Turn on the power switch. The Power indicator light and the battery level indicator light will illuminate.



Wait for a few seconds, and you will hear a slight beep, indicating that the Raspberry Pi has successfully booted.

Note: If both battery level indicator lights are off, please charge the battery. When you need extended programming or debugging sessions, you can keep the Raspberry Pi operational by inserting the USB-C cable to charge the battery simultaneously.

18650 Battery



- VCC: Battery positive terminal, here there are two sets of VCC and GND is to increase the current and reduce the resistance.
- Middle: To balance the voltage between the two cells and thus protect the battery.
- GND: Negative battery terminal.

This is a custom battery pack made by SunFounder consisting of two 18650 batteries with a capacity of 2000mAh. The connector is XH2.54 3P, which can be charged directly after being inserted into the shield.

Features

- Battery charge: 5V/2A
- Battery output: 5V/5A
- Battery capacity: 3.7V 2000mAh x 2
- Battery life: 90min
- Battery charge time: 130min

- Connector:XH2.54 3P

2.1.4 4. Setting Up Your Raspberry Pi

Setting Up with a Screen

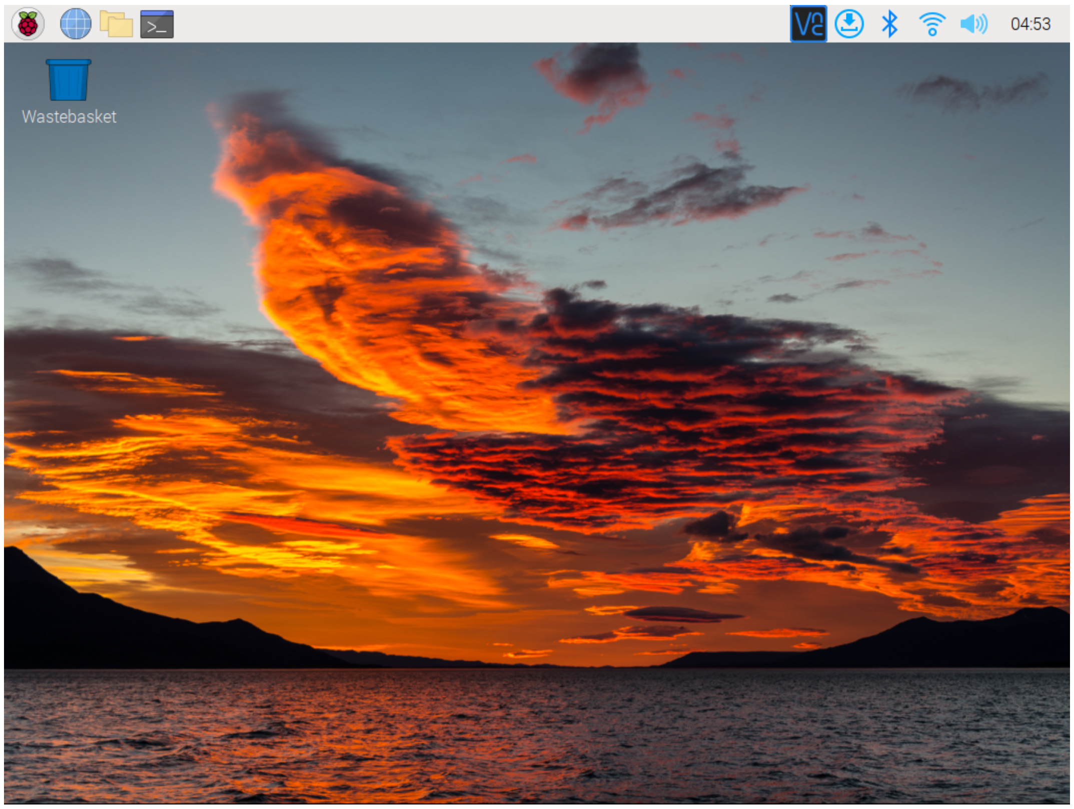
Having a screen simplifies the process of working with your Raspberry Pi.

Required Components

- Raspberry Pi 5 Model B
- Power Adapter
- Micro SD card
- Screen Power Adapter
- HDMI cable
- Screen
- Mouse
- Keyboard

Steps:

1. Connect the Mouse and Keyboard to the Raspberry Pi.
2. Use the HDMI cable to connect the screen to the Raspberry Pi's HDMI port. Ensure the screen is plugged into a power source and turned on.
3. Power the Raspberry Pi using the power adapter. The Raspberry Pi OS desktop should appear on the screen after a few seconds.



Setting Up Without a Screen

If you don't have a monitor, remote login is a viable option.

Required Components

- Raspberry Pi 5 Model B
- Power Adapter
- Micro SD card

Using SSH, you can access the Raspberry Pi's Bash shell, which is the default Linux shell. Bash offers a command-line interface for performing various tasks.

For those preferring a graphical user interface (GUI), the remote desktop feature is a convenient alternative for managing files and operations.

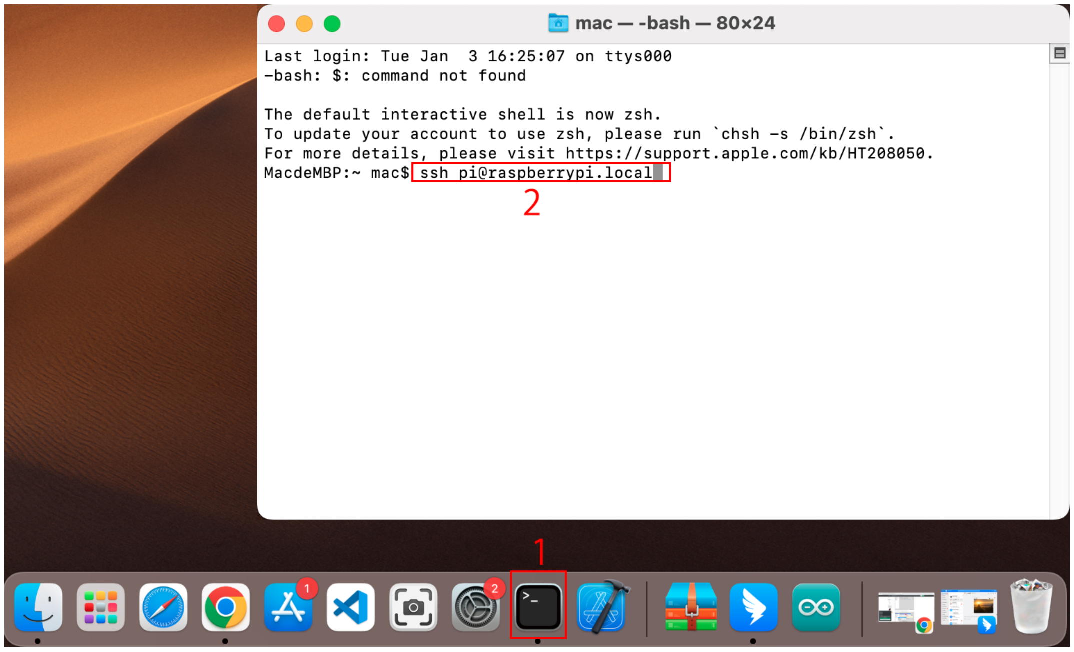
For detailed setup tutorials based on your operating system, refer to the following sections:

For Mac OS X Users

For Mac OS X users, SSH (Secure Shell) offers a secure and convenient method to remotely access and control a Raspberry Pi. This is particularly handy for working with the Raspberry Pi remotely or when it's not connected to a monitor. Using the Terminal application on a Mac, you can establish this secure connection. The process involves an SSH command incorporating the Raspberry Pi's username and hostname. During the initial connection, a security prompt will ask for confirmation of the Raspberry Pi's authenticity.

1. To connect to the Raspberry Pi, type the following SSH command:

```
ssh pi@raspberrypi.local
```



```
Last login: Tue Jan  3 16:25:07 on ttys000
-bash: $: command not found

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacdeMBP:~ mac$ ssh pi@raspberrypi.local
```

2. A security message will appear during your first login. Respond with **yes** to proceed.

```
The authenticity of host 'raspberrypi.local␣
↪(2400:2410:2101:5800:635b:f0b6:2662:8cba)' can't be established.
ED25519 key fingerprint is SHA256:oo7x3ZSgAo032wD1tE8eW0fFM/
↪kmewIvRwkBys6XRwg.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

3. Input the password for the Raspberry Pi. Be aware that the password won't display on the screen as you type, which is a standard security feature.

```
pi@raspberrypi.local's password:
Linux raspberrypi 5.15.61-v8+ #1579 SMP PREEMPT Fri Aug 26 11:16:44 BST␣
↪2022 aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

(continues on next page)

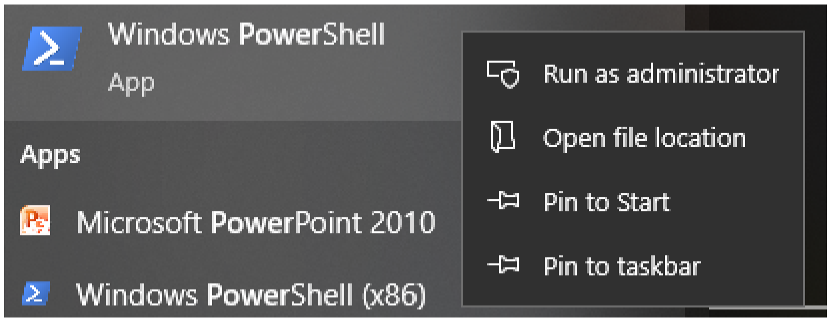
(continued from previous page)

```
Last login: Thu Sep 22 12:18:22 2022
pi@raspberrypi:~ $
```

For Windows Users

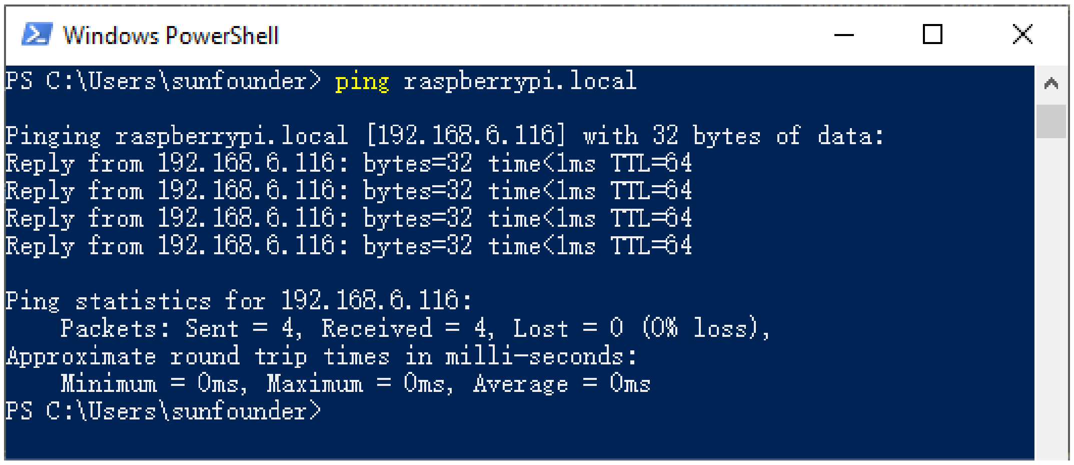
For Windows 10 or higher users, remote login to a Raspberry Pi can be achieved through the following steps:

1. Search for `powershell` in your Windows search box. Right-click on `Windows PowerShell` and select `Run as administrator`.



2. Determine your Raspberry Pi's IP address by typing `ping -4 <hostname>.local` in PowerShell.

```
ping -4 raspberrypi.local
```



```
PS C:\Users\sunfounder> ping raspberrypi.local

Pinging raspberrypi.local [192.168.6.116] with 32 bytes of data:
Reply from 192.168.6.116: bytes=32 time<1ms TTL=64
Reply from 192.168.6.116: bytes=32 time<1ms TTL=64
Reply from 192.168.6.116: bytes=32 time<1ms TTL=64
Reply from 192.168.6.116: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.6.116:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
PS C:\Users\sunfounder>
```

The Raspberry Pi's IP address will be displayed once it's connected to the network.

- If the terminal displays `Ping request could not find host pi.local. Please check the name and try again.`, verify the hostname you've entered is correct.
- If the IP address still isn't retrievable, check your network or WiFi settings on the Raspberry Pi.

3. Once the IP address is confirmed, log in to your Raspberry Pi using `ssh <username>@<hostname>.local` or `ssh <username>@<IP address>`.

```
ssh pi@raspberrypi.local
```

Warning: If an error appears stating `The term 'ssh' is not recognized as the name of a cmdlet...`, your system may not have SSH tools pre-installed. In this case, you need to manually install OpenSSH following *Install OpenSSH via Powershell*, or use a third-party tool as described in *PuTTY*.

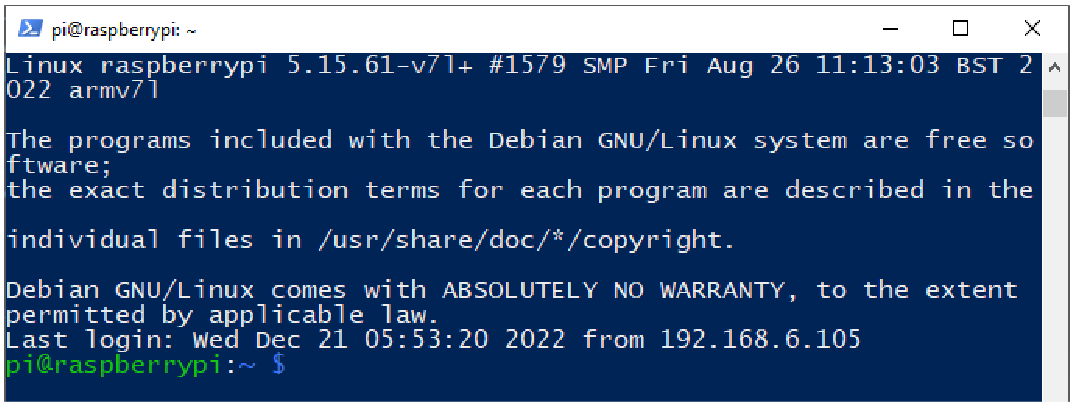
4. A security message will appear on your first login. Enter `yes` to proceed.

```
The authenticity of host 'raspberrypi.local␣
↪(2400:2410:2101:5800:635b:f0b6:2662:8cba)' can't be established.
ED25519 key fingerprint is SHA256:oo7x3ZSgAo032wD1tE8eW0fFM/
↪kmewIvRwkBys6XRwg.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

5. Enter the password you previously set. Note that the password characters won't be displayed on the screen, which is a standard security feature.

Note: The absence of visible characters when typing the password is normal. Ensure you input the correct password.

6. Once connected, your Raspberry Pi is ready for remote operations.



```
Linux raspberrypi 5.15.61-v7l+ #1579 SMP Fri Aug 26 11:13:03 BST 2
022 armv7l

The programs included with the Debian GNU/Linux system are free so
ftware;
the exact distribution terms for each program are described in the

individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Dec 21 05:53:20 2022 from 192.168.6.105
pi@raspberrypi:~ $
```

For Linux/Unix Users

1. Locate and open the **Terminal** on your Linux/Unix system.
2. Ensure your Raspberry Pi is connected to the same network. Verify this by typing *ping <hostname>.local*. For example:

```
ping raspberrypi.local
```

You should see the Raspberry Pi's IP address if it's connected to the network.

- If the terminal shows a message like `Ping request could not find host pi.local. Please check the name and try again.`, double-check the hostname you've entered.
- If you're unable to retrieve the IP address, inspect your network or WiFi settings on the Raspberry Pi.

3. Initiate an SSH connection by typing `ssh <username>@<hostname>.local` or `ssh <username>@<IP address>`. For instance:

```
ssh pi@raspberrypi.local
```

4. On your first login, you'll encounter a security message. Type `yes` to proceed.

```
The authenticity of host 'raspberrypi.local␣
↪(2400:2410:2101:5800:635b:f0b6:2662:8cba)' can't be established.
ED25519 key fingerprint is SHA256:oo7x3ZSgAo032wD1tE8eW0fFM/
↪kmewIvRwkBys6XRwg.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

5. Enter the password you previously set. Note that for security reasons, the password won't be visible as you type.

Note: It's normal for the password characters not to display in the terminal. Just ensure to enter the correct password.

6. Once you've successfully logged in, your Raspberry Pi is now connected, and you're ready to proceed to the next step.

Remote Desktop Access for Raspberry Pi

For those preferring a graphical user interface (GUI) over command-line access, the Raspberry Pi supports remote desktop functionality. This guide will walk you through setting up and using VNC (Virtual Network Computing) for remote access.

We recommend using VNC® Viewer for this purpose.

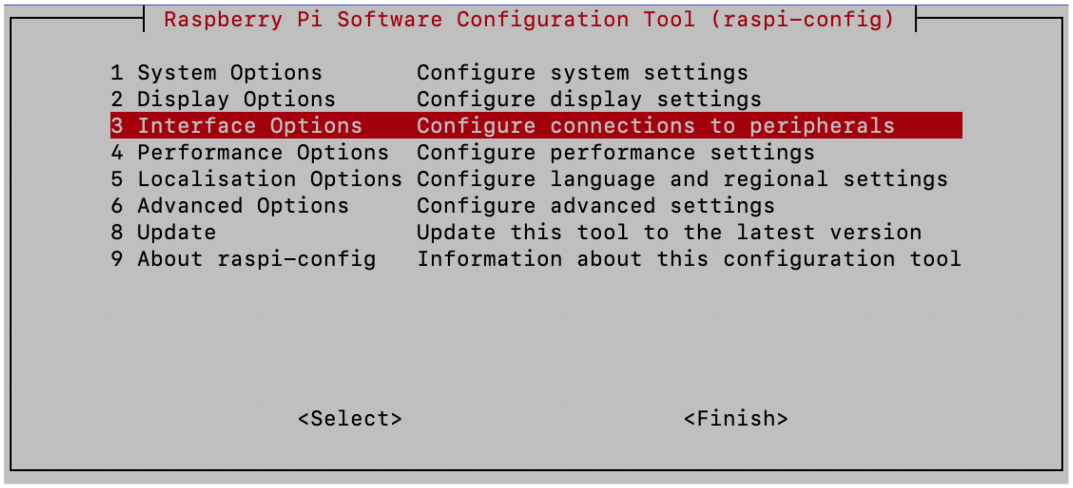
Enabling VNC Service on Raspberry Pi

VNC service comes pre-installed in the Raspberry Pi OS but is disabled by default. Follow these steps to enable it:

1. Enter the following command in the Raspberry Pi terminal:

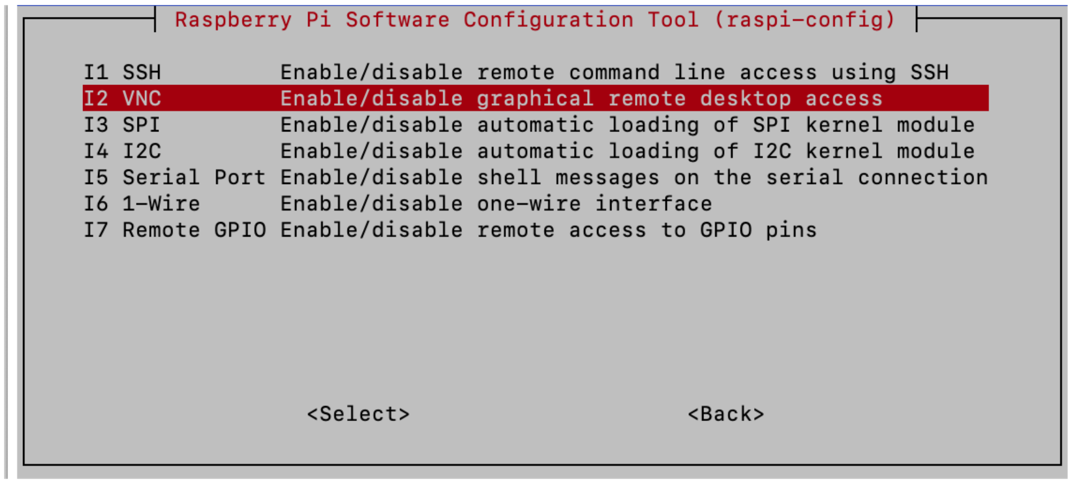
```
sudo raspi-config
```

2. Navigate to **Interfacing Options** using the down arrow key, then press **Enter**.



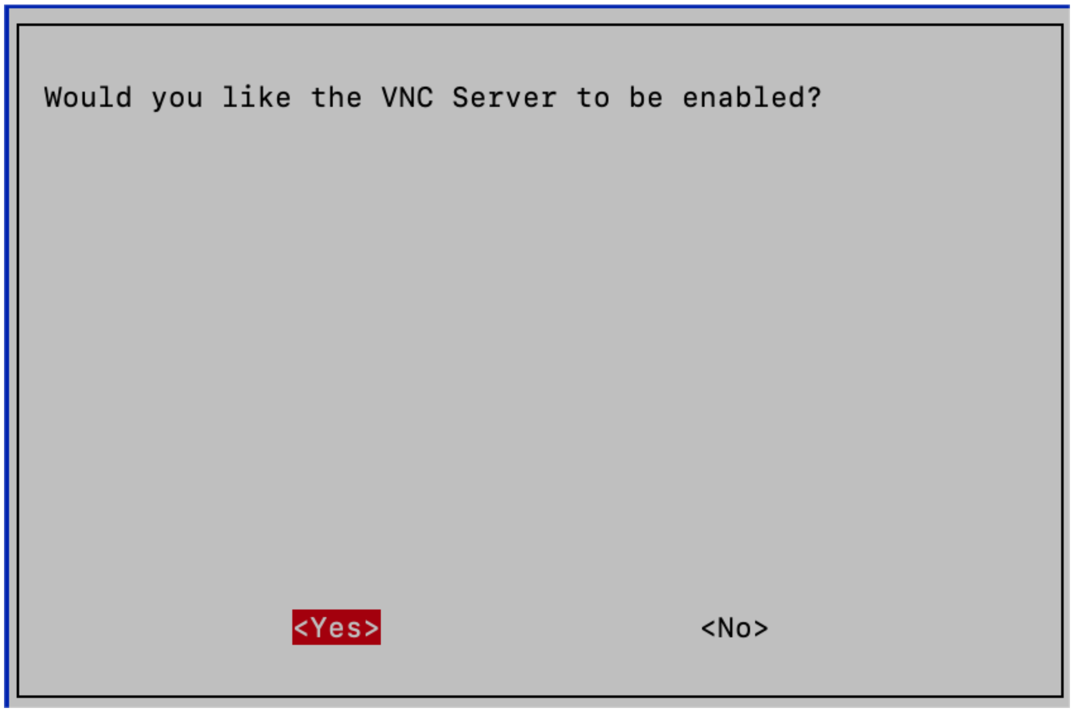
```
Raspberry Pi Software Configuration Tool (raspi-config)

1 System Options       Configure system settings
2 Display Options      Configure display settings
3 Interface Options    Configure connections to peripherals
4 Performance Options  Configure performance settings
5 Localisation Options Configure language and regional settings
6 Advanced Options     Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

            <Select>                    <Finish>
```

3. Select **VNC** from the options.



```
Raspberry Pi Software Configuration Tool (raspi-config)

I1 SSH         Enable/disable remote command line access using SSH
I2 VNC         Enable/disable graphical remote desktop access
I3 SPI         Enable/disable automatic loading of SPI kernel module
I4 I2C         Enable/disable automatic loading of I2C kernel module
I5 Serial Port Enable/disable shell messages on the serial connection
I6 1-Wire      Enable/disable one-wire interface
I7 Remote GPIO Enable/disable remote access to GPIO pins

              <Select>                      <Back>
```

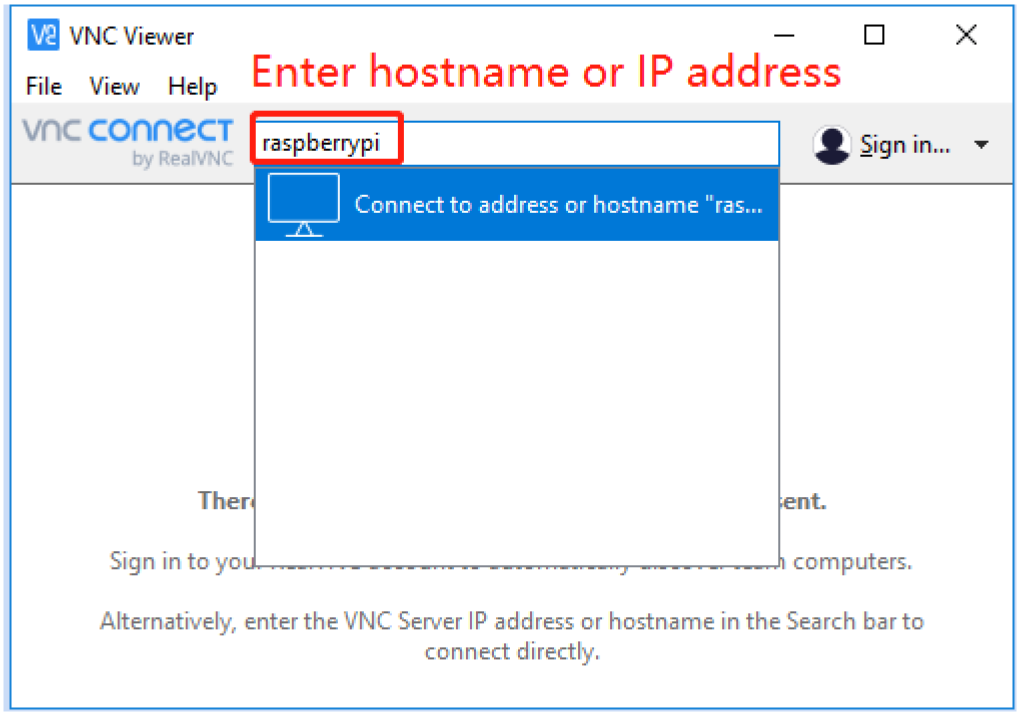
4. Use the arrow keys to choose **<Yes>** -> **<OK>** -> **<Finish>** and finalize the VNC service activation.



```
Would you like the VNC Server to be enabled?

              <Yes>                     <No>
```

Logging in via VNC Viewer

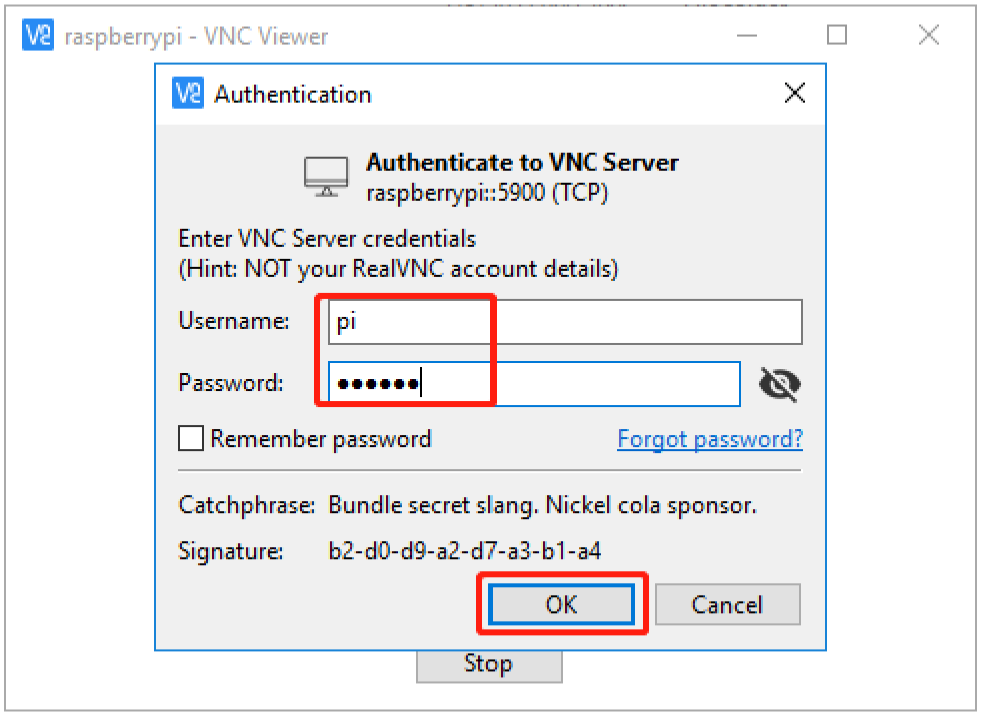
1. Download and install VNC Viewer on your personal computer.
2. Once installed, launch VNC Viewer. Enter the hostname or IP address of your Raspberry Pi and press Enter.


VNC Viewer
File View Help
Enter hostname or IP address
raspberrypi
Sign in...
Connect to address or hostname "ras...

Sign in to you... computers.

Alternatively, enter the VNC Server IP address or hostname in the Search bar to connect directly.

3. When prompted, enter your Raspberry Pi's username and password, then click **OK**.



Authenticate to VNC Server
raspberrypi::5900 (TCP)

Enter VNC Server credentials
(Hint: NOT your RealVNC account details)

Username: pi

Password: ••••••

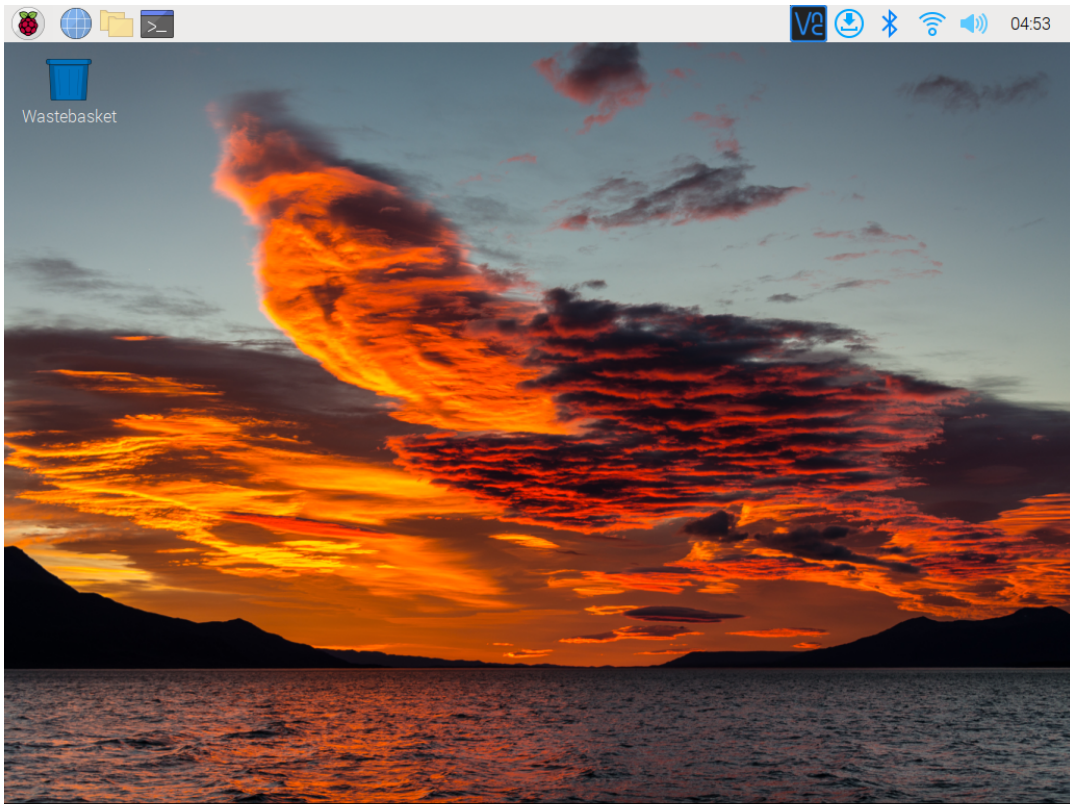
Remember password   Forgot password?

Catchphrase: Bundle secret slang. Nickel cola sponsor.

Signature: b2-d0-d9-a2-d7-a3-b1-a4

OK   Cancel

4. You'll now have access to your Raspberry Pi's desktop interface.



2.1.5 5. Install All the Modules(Important)

1. Update your system.

   Make sure you are connected to the Internet and update your system:

   ```
   sudo apt update
   sudo apt upgrade
   ```

   Note: Python3 related packages must be installed if you are installing the Lite version OS.

   ```
   sudo apt install git python3-pip python3-setuptools python3-smbus
   ```

2. Install `robot-hat` module.

   ```
   cd ~/
   git clone -b v2.0 https://github.com/sunfounder/robot-hat.git
   cd robot-hat
   sudo python3 setup.py install
   ```

3. Install `vilib` module.

   ```
   cd ~/
   git clone -b picamera2 https://github.com/sunfounder/vilib.git
   cd vilib
   sudo python3 install.py
   ```

4. Download the code.

   ```
   cd ~/
   git clone https://github.com/sunfounder/pidog.git
   ```

5. Install `pidog` module.

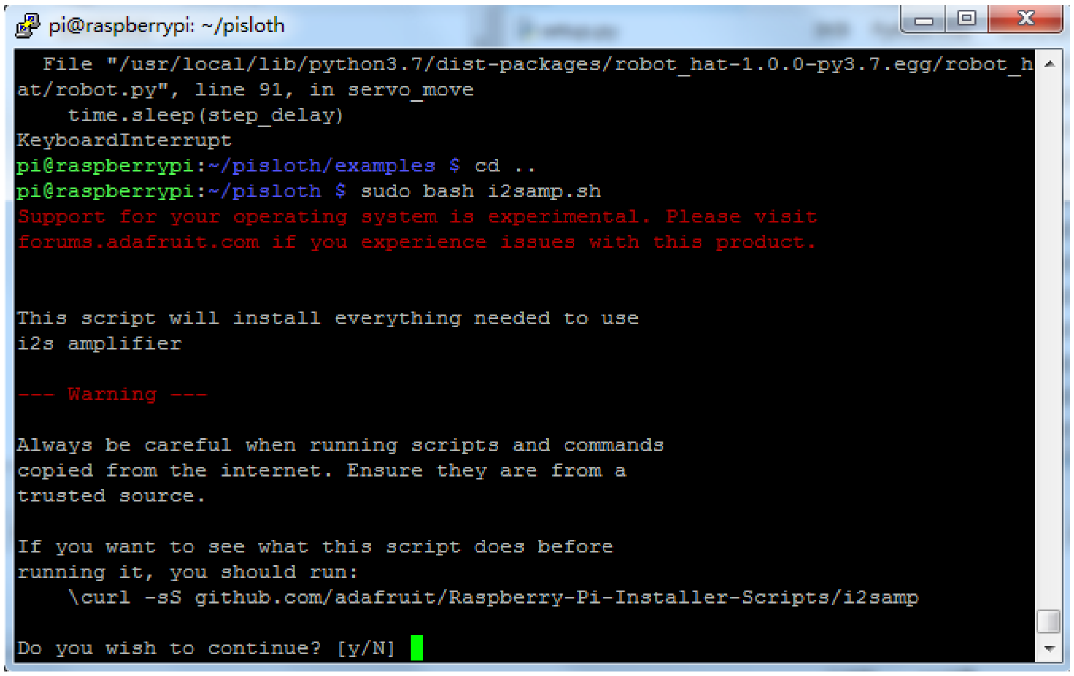
   ```
   cd pidog
   sudo python3 setup.py install
   ```

   This step will take a little time, so please be patient.

6. Run the script `i2samp.sh`.

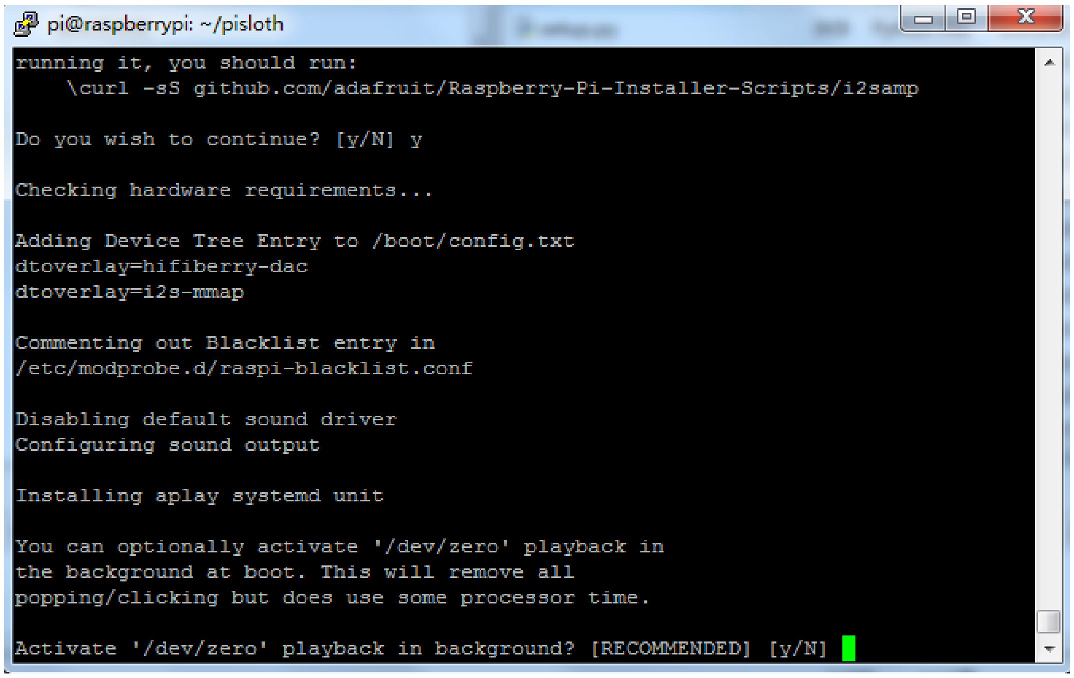
   Finally, you need to run the script `i2samp.sh` to install the components required by the i2s amplifier, otherwise the robot will have no sound.

   ```
   cd ~/pidog
   sudo bash i2samp.sh
   ```



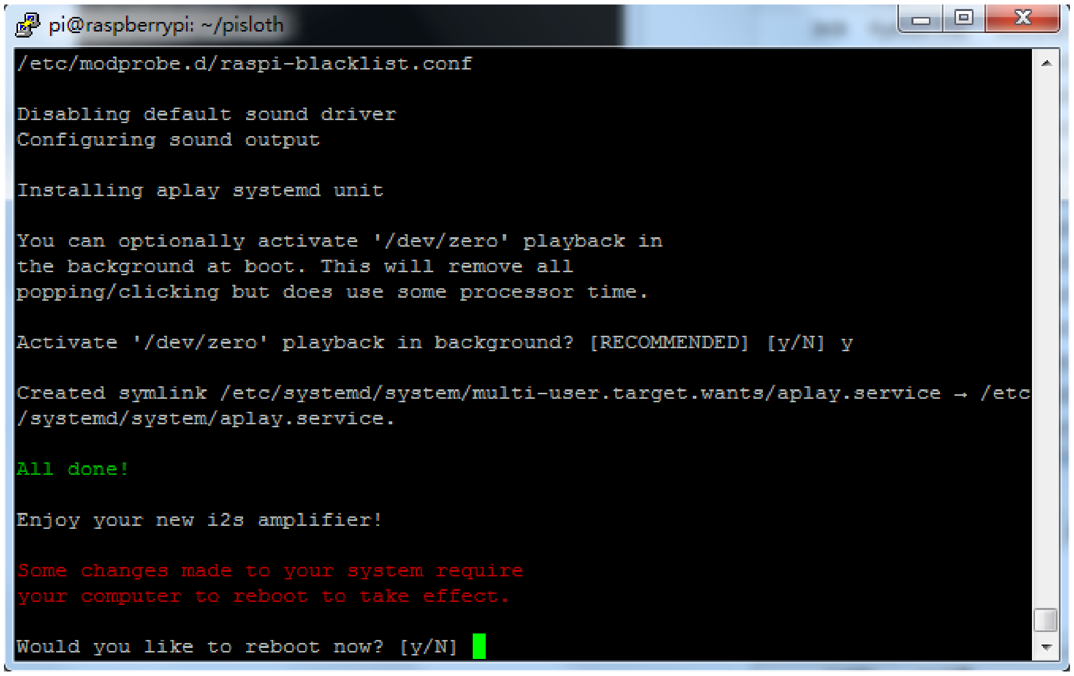
```
  File "/usr/local/lib/python3.7/dist-packages/robot_hat-1.0.0-py3.7.egg/robot_h
at/robot.py", line 91, in servo_move
    time.sleep(step_delay)
KeyboardInterrupt
pi@raspberrypi:~/pisloth/examples $ cd ..
pi@raspberrypi:~/pisloth $ sudo bash i2samp.sh
Support for your operating system is experimental. Please visit
forums.adafruit.com if you experience issues with this product.


This script will install everything needed to use
i2s amplifier

--- Warning ---

Always be careful when running scripts and commands
copied from the internet. Ensure they are from a
trusted source.

If you want to see what this script does before
running it, you should run:
    \curl -sS github.com/adafruit/Raspberry-Pi-Installer-Scripts/i2samp

Do you wish to continue? [y/N]
```

   Type `y` and press `Enter` to continue running the script.



```
running it, you should run:
    \curl -sS github.com/adafruit/Raspberry-Pi-Installer-Scripts/i2samp

Do you wish to continue? [y/N] y

Checking hardware requirements...

Adding Device Tree Entry to /boot/config.txt
dtoverlay=hifiberry-dac
dtoverlay=i2s-mmap

Commenting out Blacklist entry in
/etc/modprobe.d/raspi-blacklist.conf

Disabling default sound driver
Configuring sound output

Installing aplay systemd unit

You can optionally activate '/dev/zero' playback in
the background at boot. This will remove all
popping/clicking but does use some processor time.

Activate '/dev/zero' playback in background? [RECOMMENDED] [y/N]
```

Type `y` and press `Enter` to run `/dev/zero` in the background.



```
/etc/modprobe.d/raspi-blacklist.conf

Disabling default sound driver
Configuring sound output

Installing aplay systemd unit

You can optionally activate '/dev/zero' playback in
the background at boot. This will remove all
popping/clicking but does use some processor time.

Activate '/dev/zero' playback in background? [RECOMMENDED] [y/N] y

Created symlink /etc/systemd/system/multi-user.target.wants/aplay.service → /etc
/systemd/system/aplay.service.

All done!

Enjoy your new i2s amplifier!

Some changes made to your system require
your computer to reboot to take effect.

Would you like to reboot now? [y/N]
```

Type `y` and press `Enter` to restart the machine.

Note: If there is no sound after restarting, you may need to run the `i2samp.sh` script multiple times.

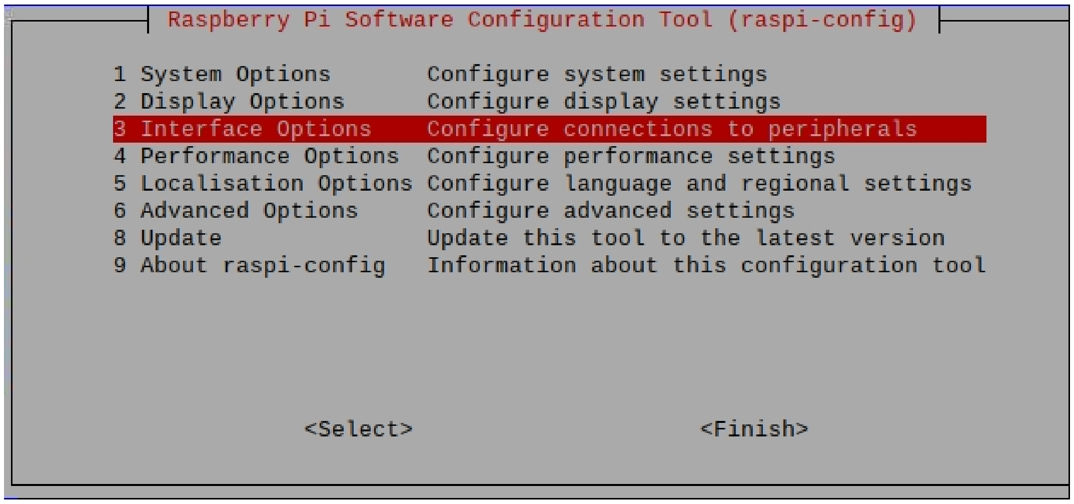
2.1.6 6. Check I2C and SPI Interface

We will be using Raspberry Pi's I2C and SPI interfaces. These interfaces should have been enabled when installing the `robot-hat` module earlier. To ensure everything is in order, let's check if they are indeed enabled.

1. Input the following command:

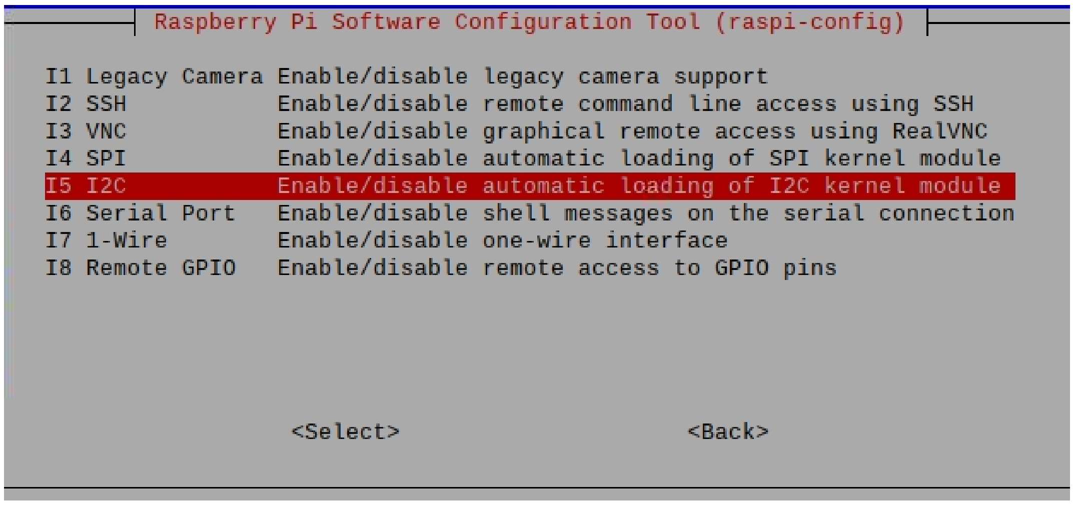
```
sudo raspi-config
```

2. Choose **Interfacing Options** by press the down arrow key on your keyboard, then press the **Enter** key.



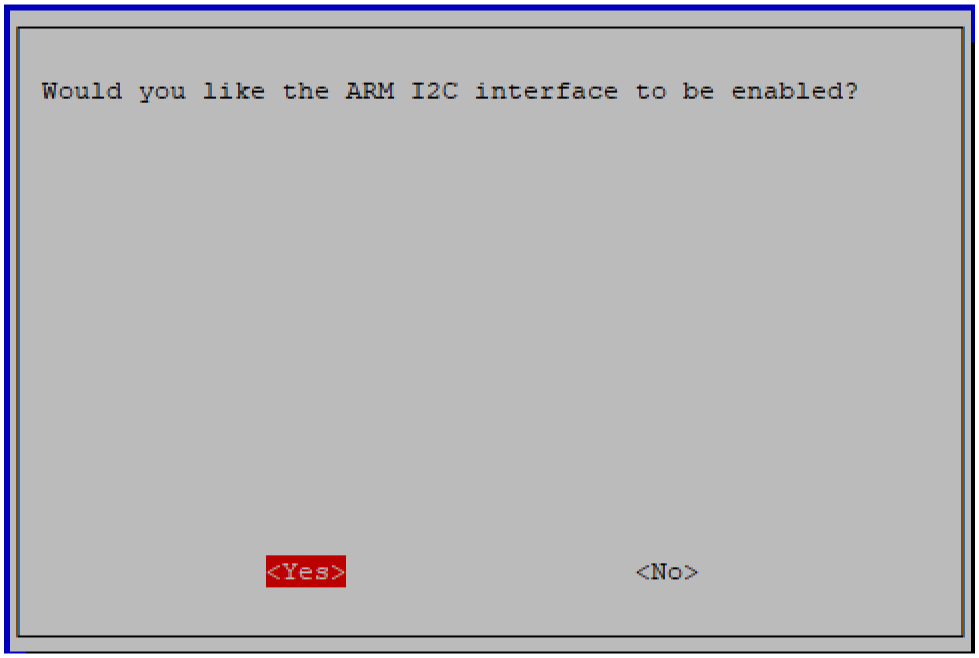
```
Raspberry Pi Software Configuration Tool (raspi-config)

1 System Options       Configure system settings
2 Display Options      Configure display settings
3 Interface Options    Configure connections to peripherals
4 Performance Options  Configure performance settings
5 Localisation Options Configure language and regional settings
6 Advanced Options     Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

              <Select>                     <Finish>
```

3. Then **I2C**.



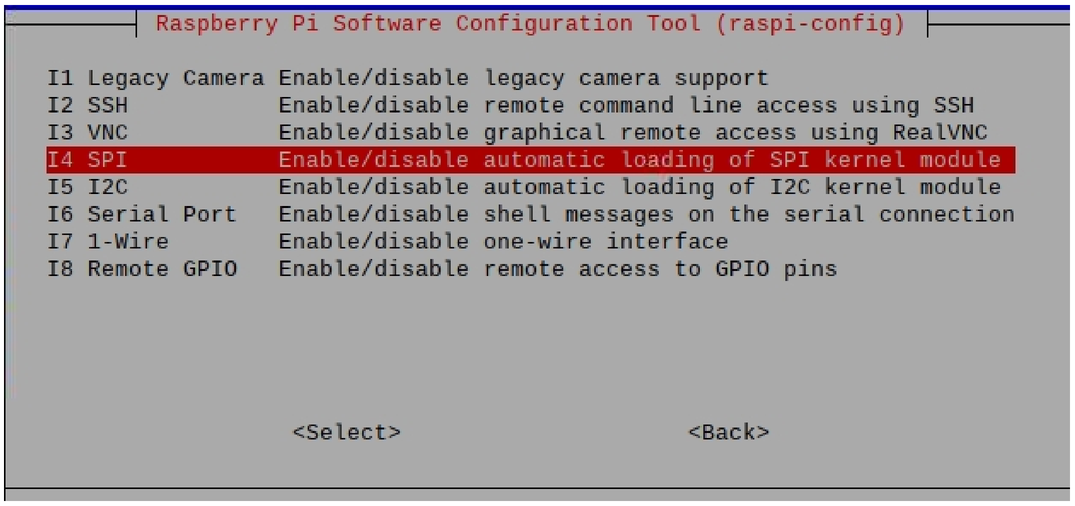
```
Raspberry Pi Software Configuration Tool (raspi-config)

I1 Legacy Camera Enable/disable legacy camera support
I2 SSH           Enable/disable remote command line access using SSH
I3 VNC           Enable/disable graphical remote access using RealVNC
I4 SPI           Enable/disable automatic loading of SPI kernel module
I5 I2C           Enable/disable automatic loading of I2C kernel module
I6 Serial Port   Enable/disable shell messages on the serial connection
I7 1-Wire        Enable/disable one-wire interface
I8 Remote GPIO   Enable/disable remote access to GPIO pins

              <Select>                     <Back>
```

4. Use the arrow keys on the keyboard to select **<Yes>** -> **<OK>** to complete the setup of the I2C.



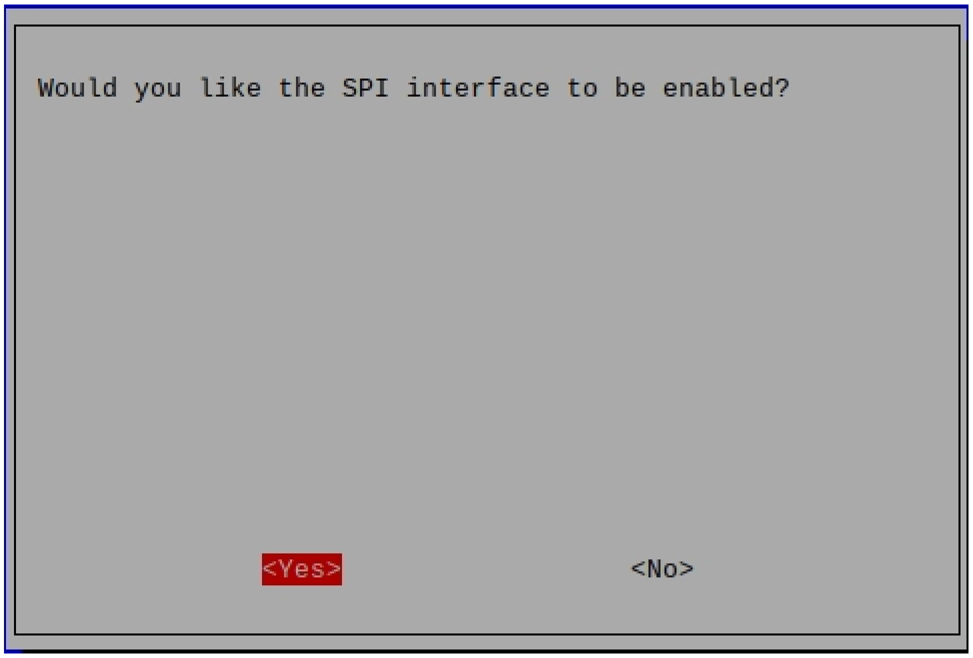
```
Would you like the ARM I2C interface to be enabled?
```

5. Go to **Interfacing Options** again and select **SPI**.



```
Raspberry Pi Software Configuration Tool (raspi-config)

I1 Legacy Camera Enable/disable legacy camera support
I2 SSH           Enable/disable remote command line access using SSH
I3 VNC           Enable/disable graphical remote access using RealVNC
I4 SPI           Enable/disable automatic loading of SPI kernel module
I5 I2C           Enable/disable automatic loading of I2C kernel module
I6 Serial Port   Enable/disable shell messages on the serial connection
I7 1-Wire        Enable/disable one-wire interface
I8 Remote GPIO   Enable/disable remote access to GPIO pins
```

6. Use the arrow keys on the keyboard to select **<Yes>** -> **<OK>** to complete the setup of the SPI.



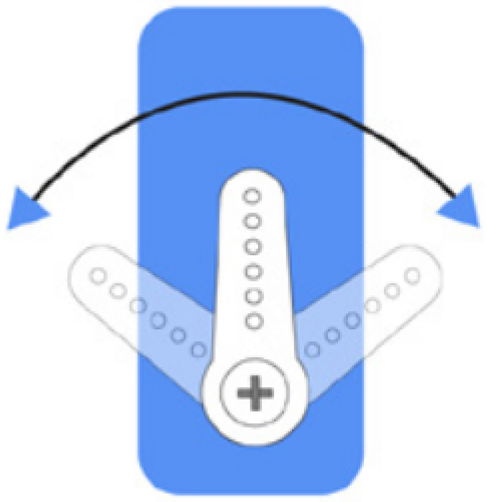
```
Would you like the SPI interface to be enabled?
```

2.1.7 7. Servo Adjust(Importtant)

The angle range of the servo is -90~90, but the angle set at the factory is random, maybe 0°, maybe 45°; if we assemble it with such an angle directly, it will lead to a chaotic state after the robot runs the code, or worse, it will cause the servo to block and burn out.

So here we need to set all the servo angles to 0° and then install them, so that the servo angle is in the middle, no matter which direction to turn.

1. To ensure that the servo has been properly set to 0°, first insert the servo arm into the servo shaft and then gently rotate the rocker arm to a different angle. This servo arm is just to allow you to clearly see that the servo is rotating.

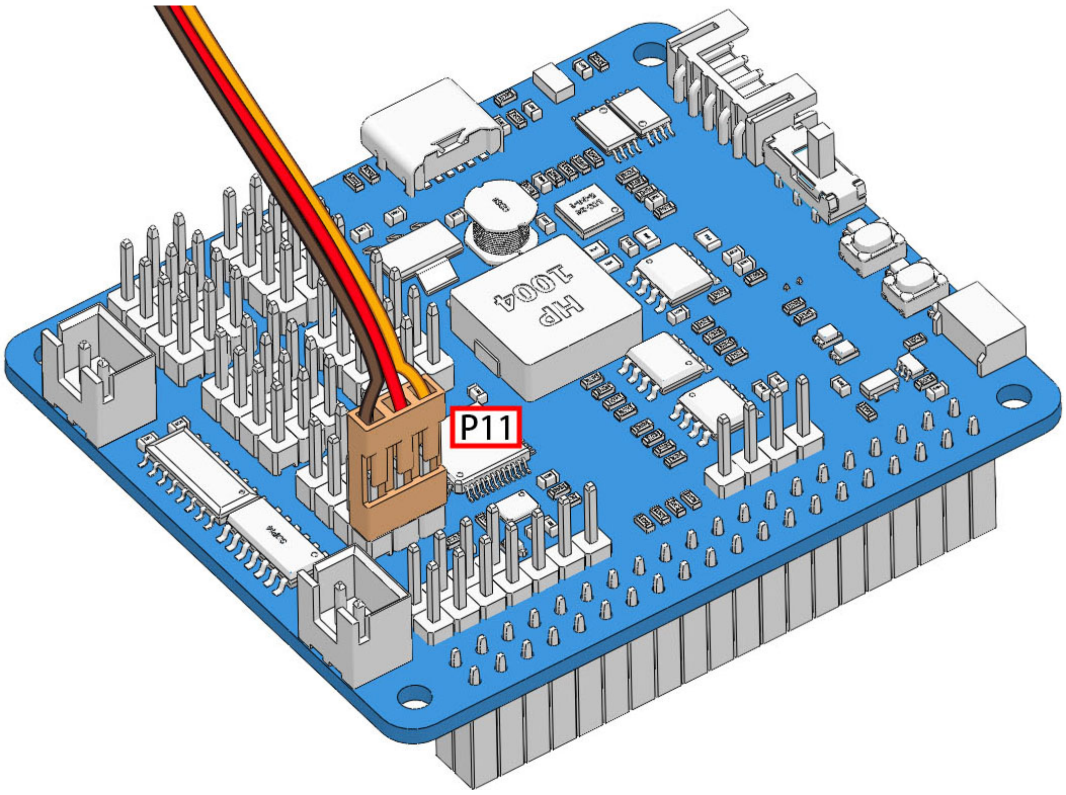


2. Now, run `servo_zeroing.py` in the `examples/` folder.

```
cd ~/pidog/examples
sudo python3 servo_zeroing.py
```

Note: If you get an error, try re-enabling the Raspberry Pi's I2C port, see: *6. Check I2C and SPI Interface*.

3. Next, plug the servo cable into the P11 port as follows, at the same time you will see the servo arm rotate to a position(This is the 0° position, which is a random location and may not be vertical or parallel.).



4. Now, remove the servo arm, ensuring the servo wire remains connected, and do not turn off the power. Then continue the assembly following the paper instructions.

Note:

- Do not unplug this servo cable before fixing it with the servo screw, you can unplug it after fixing it.
- Do not rotate the servo while it is powered on to avoid damage; if the servo shaft is not inserted at the right angle, pull the servo out and reinsert it.
- Before assembling each servo, you need to plug the servo cable into PWM pin and turn on the power to set its angle to 0°.

Video

In our assembly video from **3:40 to 7:23**, there is also a detailed tutorial for this chapter. You can follow the video instructions directly.

As soon as the assembly is completed, you need to calibrate the PiDog to prevent it from damaging the servo if there is a slight deviation in the assembly.

2.2 2. Calibrate the PiDog

Introduction

Calibrating your PiDog is an essential step to ensure its stable and efficient operation. This process helps correct any imbalances or inaccuracies that might have arisen during assembly or from structural issues. Follow these steps carefully to ensure your PiDog walks steadily and performs as expected.

But if the deviation angle is too big, you still have to go back to *7. Servo Adjust(Importtant)* to set the servo angle to 0°, and then follow the instructions to reassemble the PiDog.

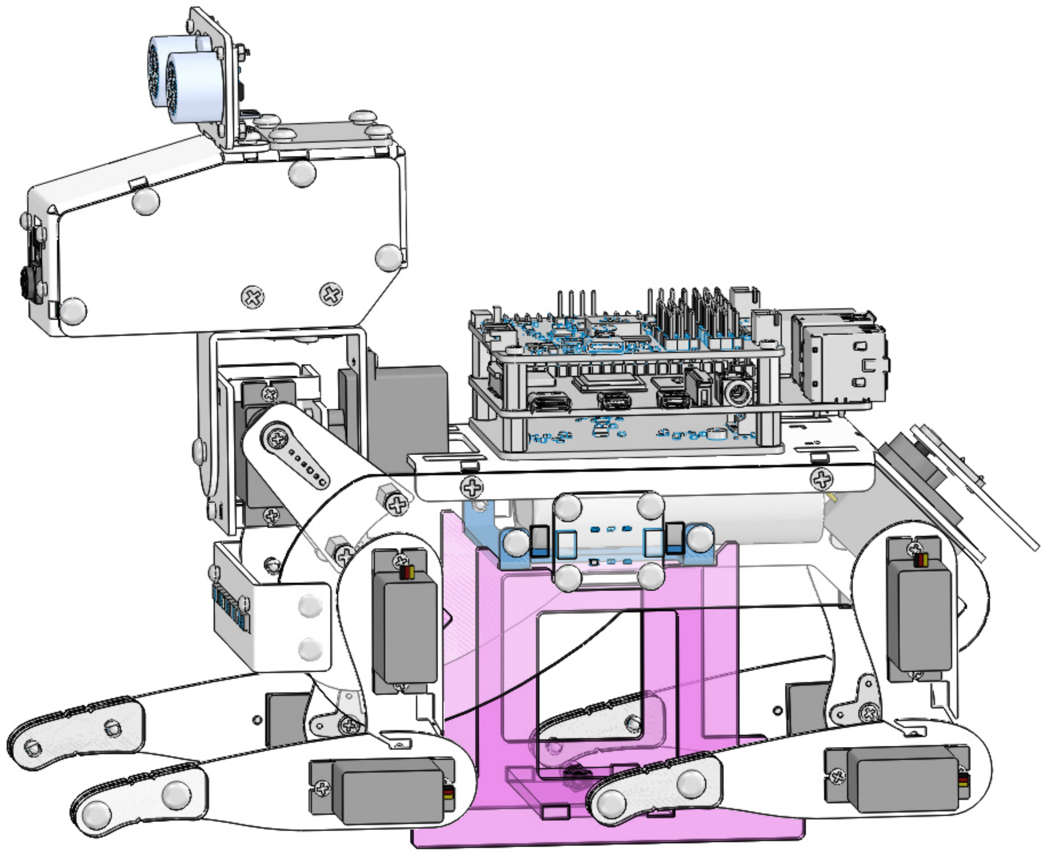
Calibrate Video

For a comprehensive guide, refer to the full calibration video. It provides a visual step-by-step process to accurately calibrate your PiDog.

Steps

The specific steps are as follows:

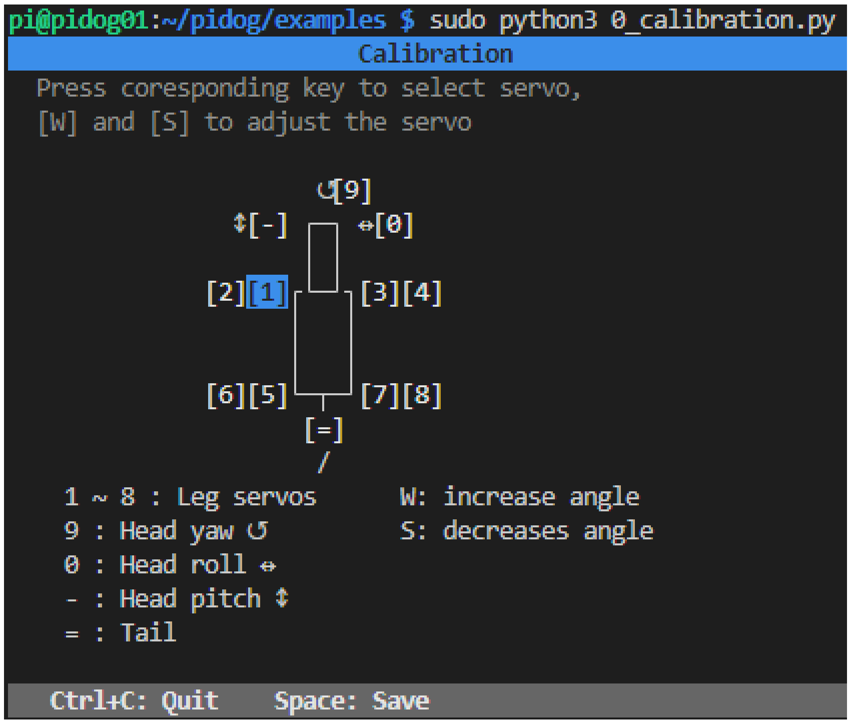
1. Put the PiDog on the base.



2. Navigate to the PiDog examples directory and run the `0_calibration.py` script.

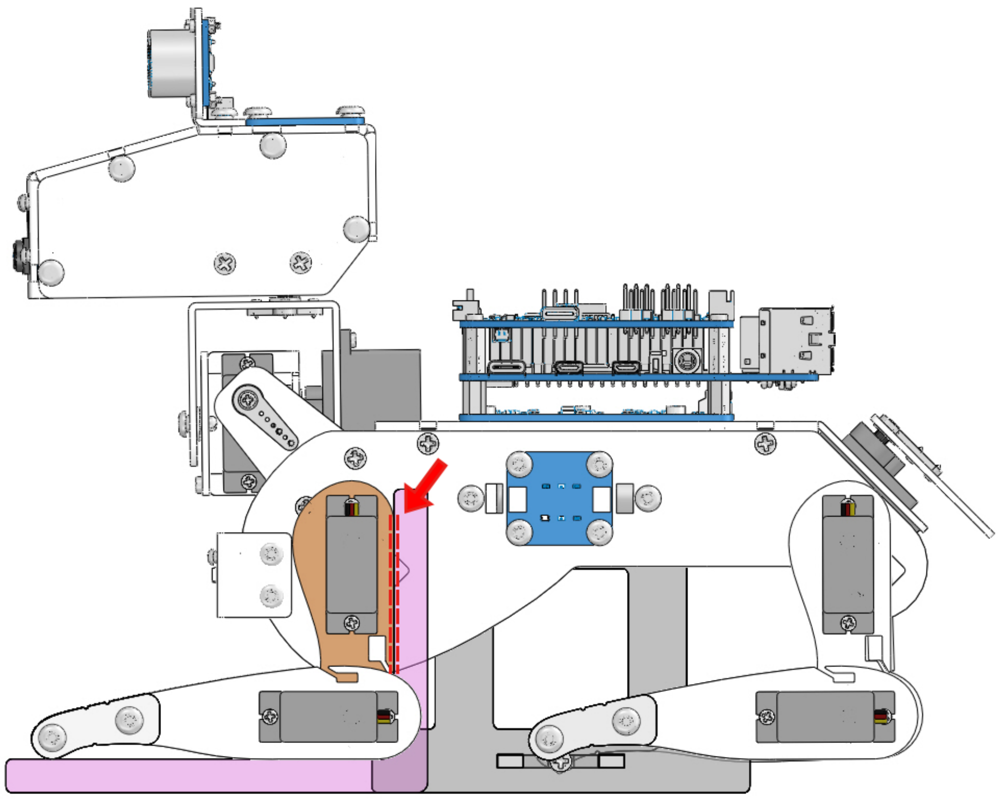
```
cd ~/pidog/examples
sudo python3 0_calibration.py
```

Upon running the script, a user interface will appear in your terminal.

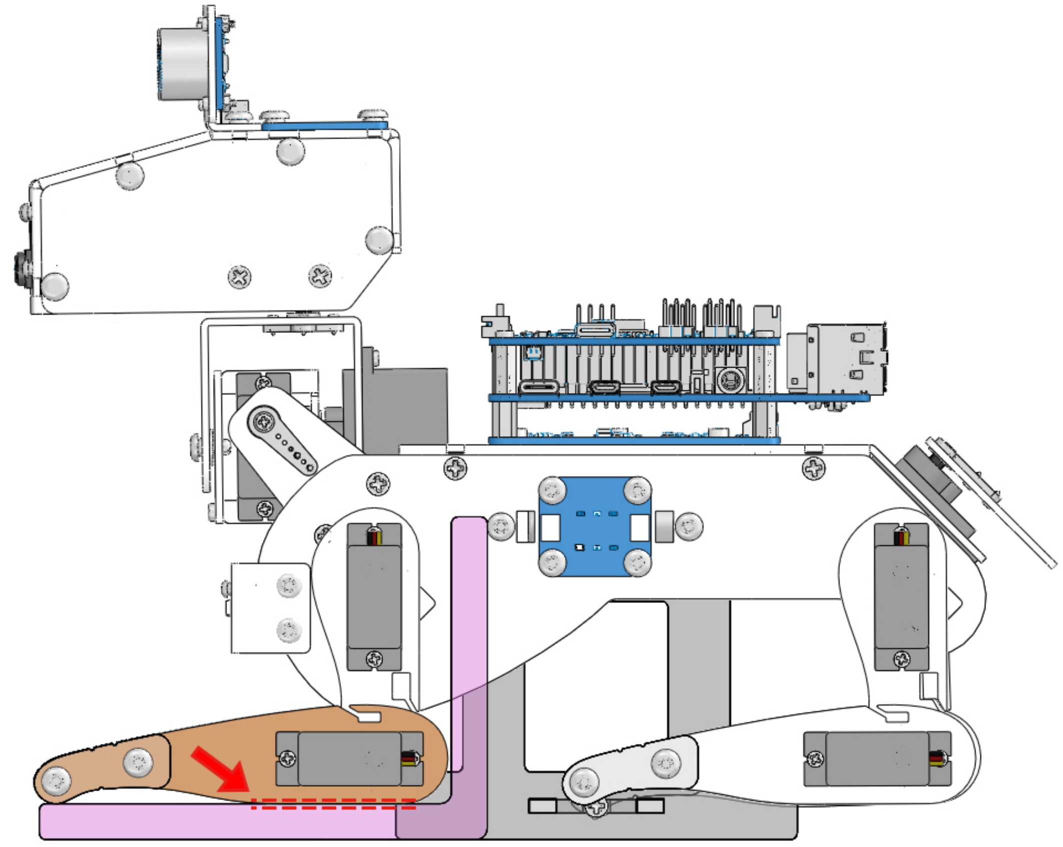


```
pi@pidog01:~/pidog/examples $ sudo python3 0_calibration.py
                          Calibration
  Press coresponding key to select servo,
  [W] and [S] to adjust the servo

                         ↺[9]
                 ⇕[-]         ⇔[0]

               [2][1]         [3][4]


               [6][5]         [7][8]
                        [=]
                         /
    1 ~ 8 : Leg servos        W: increase angle
    9 : Head yaw ↺            S: decreases angle
    0 : Head roll ⇔
    - : Head pitch ⇕
    = : Tail

   Ctrl+C: Quit     Space: Save
```

3. Position the **Calibration Ruler** (Acrylic C) as shown in the provided image. In the terminal, press `1`, followed by `w` and `s` keys to align the edges as indicated in the image.



4. Reposition the **Calibration Ruler** (Acrylic C) as illustrated in the next image. Press `2` in the terminal, then use `w` and `s` to align the edges as shown.



5. Repeat the calibration process for the remaining servos (3 to 8). Ensure all four legs of the PiDog are calibrated.

You can also have PiDog achieve the following project effects.

2.3 3. Fun Python Projects

Here, we delve into an exciting assortment of projects that showcase the versatility and capabilities of the PiDog. From the basics of setting a wakeup routine in *1. Wake Up* to the advanced dynamics of ball tracking in *13. Ball Track*, each project offers a unique glimpse into the world of Python programming for robotics. Whether you're keen on making your PiDog patrol an area, respond to commands, execute pushups, or even howl on command, there's a project tailored for you. Furthermore, for those looking to extend their PiDog's capabilities to computer interfaces, we have tutorials on keyboard and app control as well. Dive in, and embark on a journey of discovery and fun with these hands-on Python projects for your PiDog!

2.3.1 1. Wake Up

This is PiDog's first project. It will wake your PiDog from a deep sleep.

Run the Code

```
cd ~/pidog/examples
sudo python3 1_wake_up.py
```

After the code is executed, PiDog will perform the following actions in sequence:

Stretch, twist, sit, wag its tail, pant.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep
from preset_actions import pant
from preset_actions import body_twisting

my_dog = Pidog(head_init_angles=[0, 0, -30])
sleep(1)

def wake_up():
    # stretch
    my_dog.rgb_strip.set_mode('listen', color='yellow', bps=0.6, brightness=0.8)
    my_dog.do_action('stretch', speed=50)
    my_dog.head_move([[0, 0, 30]]*2, immediately=True)
    my_dog.wait_all_done()
    sleep(0.2)
    body_twisting(my_dog)
    my_dog.wait_all_done()
    sleep(0.5)
    my_dog.head_move([[0, 0, -30]], immediately=True, speed=90)
    # sit and wag_tail
    my_dog.do_action('sit', speed=25)
    my_dog.wait_legs_done()
    my_dog.do_action('wag_tail', step_count=10, speed=100)
    my_dog.rgb_strip.set_mode('breath', color=[245, 10, 10], bps=2.5, brightness=0.8)
    pant(my_dog, pitch_comp=-30, volume=80)
    my_dog.wait_all_done()
    # hold
    my_dog.do_action('wag_tail', step_count=10, speed=30)
    my_dog.rgb_strip.set_mode('breath', 'pink', bps=0.5)
    while True:
        sleep(1)

if __name__ == "__main__":
    try:
```

(continues on next page)

(continued from previous page)

```
        wake_up()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        my_dog.close()
```

2.3.2 2. Function Demonstration

This project shows you all of PiDog's usual actions and sounds.

You can make PiDog make actions or make sounds by entering the serial number.

The motion/sound effects currently included in this example are listed below.

Actions:	Sound Effect:
1.stand	16.angry
2.sit	17.confused_1
3.lie	18.confused_2
4.lie_with_hands_out	19.confused_3
5.trot	20.growl_1
6.forward	21.growl_2
7.backward	22.howling
8.turn_left	23.pant
9.turn_right	24.single_bark_1
10.doze_off	25.single_bark_2
11.stretch	26.snoring
12.pushup	27.woohoo
13.shake_head	
14.tilting_head	
15.wag_tail	

Run the Code

```
cd ~/pidog/examples
sudo python3 2_function_demonstration.py
```

After running this example, you input `1` and press `ENTER`, PiDog will stand; input `2`, PiDog will sit down; input `27`, PiDog will issue "woohoo~ ".

Press `Ctrl+C` to exit the program.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from time import sleep
```

(continues on next page)

(continued from previous page)

```
from pidog import Pidog
import os
import curses
import curses_utils

# init pidog
# ======================================
my_dog = Pidog()
sleep(0.5)

# global variables
# ======================================
actions = [
    # name, head_pitch_adjust(-1, use last_pitch), speed
    ['stand', 0, 50],
    ['sit', -30, 50],
    ['lie', 0, 20],
    ['lie_with_hands_out', 0,  20],
    ['trot', 0, 95],
    ['forward', 0, 98],
    ['backward', 0, 98],
    ['turn_left', 0, 98],
    ['turn_right', 0, 98],
    ['doze_off', -30, 90],
    ['stretch', 20, 20],
    ['push_up', -30, 50],
    ['shake_head', -1, 90],
    ['tilting_head', -1, 60],
    ['wag_tail', -1, 100],
]
actions_len = len(actions)

sound_effects = []
# change working directory
abspath = os.path.abspath(os.path.dirname(__file__))
# print(abspath)
os.chdir(abspath)
for name in os.listdir('../sounds'):
    sound_effects.append(name.split('.')[0])
sound_effects.sort()
sound_len = len(sound_effects)
# limit sound quantity
if sound_len > actions_len:
    sound_len = actions_len
    sound_effects = sound_effects[:actions_len]

last_index = 0
last_display_index = 0
exit_flag = False
last_head_pitch = 0

STANDUP_ACTIONS = ['trot', 'forward', 'backward', 'turn_left', 'turn_right']
```

(continues on next page)

(continued from previous page)

```python
# define pad size
# ======================================
curses_utils.PAD_Y = 22
curses_utils.PAD_X = 70

# display fuctions
# ======================================
def display_head(subpad):
    title = "Function Demonstration"
    tip1 = "Input Function number to see how it goes."
    tip2 = "Actions will repeat 10 times."
    type_name_1 = "Actions:"
    type_name_2 = "Sound Effect:"
    tip3 = "(need to run with sudo)"

    curses_utils.clear_line(subpad, 0, color=curses_utils.BLACK_BLUE)
    subpad.addstr(0, 2, title, curses_utils.BLACK_BLUE | curses.A_BOLD)
    subpad.addstr(1, 2, tip1, curses_utils.GRAY)
    subpad.addstr(2, 2, tip2, curses_utils.GRAY)
    curses_utils.clear_line(subpad, 3, color=curses_utils.WHITE_GRAY)
    subpad.addstr(3, 2, type_name_1, curses_utils.WHITE_GRAY)
    subpad.addstr(3, 30, type_name_2, curses_utils.WHITE_GRAY)
    subpad.addstr(3, 31+len(type_name_2), tip3, curses_utils.YELLOW_GRAY)

def display_selection(subpad, index):
    global last_display_index
    # reset last selection
    if last_display_index > actions_len + sound_len-1 or last_display_index < 0:
        last_display_index = 0
    if last_display_index != index:
        if last_display_index < actions_len:
            subpad.addstr(last_display_index, 2, f"{last_display_index+1}. {actions[last_
↪display_index][0]}", curses_utils.LIGHT_GRAY)
        else:
            sound_index = last_display_index-actions_len
            subpad.addstr(sound_index, 30, f"{last_display_index+1}. {sound_
↪effects[sound_index]}", curses_utils.LIGHT_GRAY)
        last_display_index = index
    # highlight currernt selection
    if index > actions_len + sound_len-1 or index < 0:
        pass
    elif index < actions_len:
        subpad.addstr(index, 2, f"{index+1}. {actions[index][0]}", curses_utils.WHITE_
↪BLUE)
    else:
        sound_index = index-actions_len
        subpad.addstr(sound_index, 30, f"{index+1}. {sound_effects[sound_index]}",␣
↪curses_utils.WHITE_BLUE)

def display_actions(subpad):
    for i in range(actions_len):
```

(continues on next page)

(continued from previous page)

```
        subpad.addstr(i, 2, f"{i+1}. {actions[i][0]}", curses_utils.LIGHT_GRAY)
    for i in range(sound_len):
        subpad.addstr(i, 30, f"{i+actions_len+1}. {sound_effects[i]}", curses_utils.
↪LIGHT_GRAY)

def display_bottom(subpad):
    curses_utils.clear_line(subpad, 0, color=curses_utils.WHITE_GRAY)
    subpad.addstr(0, 0, "Enter function number: ", curses_utils.WHITE_GRAY)
    subpad.addstr(0, curses_utils.PAD_X-16, "Ctrl^C to quit", curses_utils.WHITE_GRAY)


def do_function(index):
    global last_index, last_head_pitch
    my_dog.body_stop()
    if index < 0:
        return
    if index < actions_len:
        name, head_pitch_adjust, speed = actions[index]
        # If last action is push_up, then lie down first
        if last_index < len(actions) and actions[last_index][0] in ('push_up'):
            last_head_pitch = 0
            my_dog.do_action('lie', speed=60)
        # If this action is trot, forward, turn left, turn right and backward, and, last␣
↪action is not, then stand up
        if name in STANDUP_ACTIONS and last_index < len(actions) and actions[last_
↪index][0] not in STANDUP_ACTIONS:
            last_head_pitch = 0
            my_dog.do_action('stand', speed=60)
        if head_pitch_adjust != -1:
            last_head_pitch = head_pitch_adjust
        my_dog.head_move_raw([[0, 0, last_head_pitch]], immediately=False, speed=60)
        my_dog.do_action(name, step_count=10, speed=speed, pitch_comp=last_head_pitch)
        last_index = index
    elif index < actions_len + sound_len:
        my_dog.speak(sound_effects[index - len(actions)], volume=80)
        last_index = index

def main(stdscr):
    # reset screen
    stdscr.clear()
    stdscr.move(4, 0)
    stdscr.refresh()

    # disable cursor
    curses.curs_set(0)

    # init color
    curses.start_color()
    curses.use_default_colors()
    curses_utils.init_preset_colors()
    curses_utils.init_preset__color_pairs()
```

(continues on next page)

(continued from previous page)

```
    # init pad
    pad = curses.newpad(curses_utils.PAD_Y, curses_utils.PAD_X)

    # init subpad
    head_pad = pad.subpad(4, curses_utils.PAD_X, 0, 0)
    selection_pad = pad.subpad(actions_len, curses_utils.PAD_X, 4, 0)
    bottom_pad = pad.subpad(1, curses_utils.PAD_X, actions_len+4, 0)
    # add content to a
    display_head(head_pad)
    display_actions(selection_pad)
    display_head(head_pad)
    curses_utils.pad_refresh(pad)
    curses_utils.pad_refresh(selection_pad)

    # for i in range(2):
    #     for i in range(30):
    #         display_selection(selection_pad, i)
    #         curses_utils.pad_refresh(selection_pad)
    #         sleep(0.1)

    # enable cursor and echo
    curses.curs_set(0)
    curses.echo()

    while True:
        # draw bottom bar
        display_bottom(bottom_pad)
        curses_utils.pad_refresh(bottom_pad)
        # reset cursor
        stdscr.move(actions_len+4, 23)
        stdscr.refresh()
        # red key
        key = stdscr.getstr()
        try:
            index = int(key) - 1
        except ValueError:
            index = -1
        # display selection
        display_selection(selection_pad, index)
        curses_utils.pad_refresh(selection_pad)
        # do fuction
        do_function(index)

        sleep(0.2)

if __name__ == "__main__":
    try:
        curses.wrapper(main)
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
```

(continues on next page)

(continued from previous page)

```
    finally:
        my_dog.close()
```

2.3.3 3. Patrol

In this project, PiDog makes a vivid behavior: patrolling.

PiDog will walk forward, if there is an obstacle in front of it, it will stop and bark.

Run the Code

```
cd ~/pidog/examples
sudo python3 3_patrol.py
```

After running this example, PiDog will wag its tail, scan left and right, and walk forward.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
import time
from pidog import Pidog
from preset_actions import bark

t = time.time()
my_dog = Pidog()
my_dog.do_action('stand', speed=80)
my_dog.wait_all_done()
time.sleep(.5)

DANGER_DISTANCE = 15

stand = my_dog.legs_angle_calculation([[0, 80], [0, 80], [30, 75], [30, 75]])

def patrol():
    distance = round(my_dog.ultrasonic.read_distance(), 2)
    print(f"distance: {distance} cm", end="", flush=True)

    # danger
    if distance < DANGER_DISTANCE:
        print("\033[0;31m DANGER !\033[m")
        my_dog.body_stop()
        head_yaw = my_dog.head_current_angles[0]
        # my_dog.rgb_strip.set_mode('boom', 'red', bps=2)
        my_dog.rgb_strip.set_mode('bark', 'red', bps=2)
        my_dog.tail_move([[0]], speed=80)
        my_dog.legs_move([stand], speed=70)
        my_dog.wait_all_done()
```

(continues on next page)

(continued from previous page)

```
        time.sleep(0.5)
        bark(my_dog, [head_yaw, 0, 0])

        while distance < DANGER_DISTANCE:
            distance = round(my_dog.ultrasonic.read_distance(), 2)
            if distance < DANGER_DISTANCE:
                print(f"distance: {distance} cm \033[0;31m DANGER !\033[m")
            else:
                print(f"distance: {distance} cm", end="", flush=True)
            time.sleep(0.01)
    # safe
    else:
        print("")
        my_dog.rgb_strip.set_mode('breath', 'white', bps=0.5)
        my_dog.do_action('forward', step_count=2, speed=98)
        my_dog.do_action('shake_head', step_count=1, speed=80)
        my_dog.do_action('wag_tail', step_count=5, speed=99)


if __name__ == "__main__":
    try:
        while True:
            patrol()
            time.sleep(0.01)
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        my_dog.close()
```

2.3.4 4. Response

In this project, PiDog will interact with you in an interesting way.

If you reach out and grab PiDog's head from the front, it will bark vigilantly.

But if you reach out from behind it and pet its head, it will enjoy it very much.

Run the Code

```
cd ~/pidog/examples
sudo python3 4_response.py
```

After running this example, PiDog's ultrasonic module will detect whether there is an obstacle ahead, If it detects your hand, it makes the breathing light glow red, takes a step back, and barks.

At the same time, the touch sensor will also work. If the touch sensor is stroked (not just touched), PiDog will shake its head, wag its tail, and show a comfortable look.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like pidog\examples. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep
from math import sin
from preset_actions import bark_action

my_dog = Pidog()
sleep(0.1)

def lean_forward():
    my_dog.speak('angry', volume=80)
    bark_action(my_dog)
    sleep(0.2)
    bark_action(my_dog)
    sleep(0.8)
    bark_action(my_dog)

def head_nod(step):
    y = 0
    r = 0
    p = 30
    angs = []
    for i in range(20):
        r = round(10*sin(i*0.314), 2)
        p = round(20*sin(i*0.314) + 10, 2)
        angs.append([y, r, p])

    my_dog.head_move(angs*step, immediately=False, speed=80)

def alert():
    my_dog.do_action('stand', step_count=1, speed=90)
    my_dog.rgb_strip.set_mode('breath', color='pink', bps=1, brightness=0.8)
    while True:
        print(
            f'distance.value: {round(my_dog.ultrasonic.read_distance(), 2)} cm, touch
↪{my_dog.dual_touch.read()}')
        # alert
        if my_dog.ultrasonic.read_distance() < 15 and my_dog.ultrasonic.read_distance() >
↪ 1:
            my_dog.head_move([[0, 0, 0]], immediately=True, speed=90)
            my_dog.tail_move([[0]], immediately=True, speed=90)
            my_dog.rgb_strip.set_mode('bark', color='red', bps=2, brightness=0.8)
            my_dog.do_action('backward', step_count=1, speed=98)
            my_dog.wait_all_done()
            lean_forward()
            while len(my_dog.legs_action_buffer) > 0:
                sleep(0.1)
            my_dog.do_action('stand', step_count=1, speed=90)
            sleep(0.5)
        # relax
```

(continues on next page)

(continued from previous page)

```
        if my_dog.dual_touch.read() != 'N':
            if len(my_dog.head_action_buffer) < 2:
                head_nod(1)
                my_dog.do_action('wag_tail', step_count=10, speed=80)
                my_dog.rgb_strip.set_mode('listen', color="#8A2BE2", bps=0.35,␣
↪brightness=0.8)
        # calm
        else:
            my_dog.rgb_strip.set_mode('breath', color='pink', bps=1, brightness=0.8)
            my_dog.tail_stop()
        sleep(0.2)

if __name__ == "__main__":
    try:
        alert()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        my_dog.close()
```

2.3.5 5. Rest

PiDog will doze off on the ground, and when it hears sounds around it, it will stand up in confusion to see who woke it up.

Run the Code

```
cd ~/pidog/examples
sudo python3 5_rest.py
```

After the program runs, PiDog will get down on the ground, shake its head and tail as if dozing off. At the same time, its sound direction sensor module is working. If PiDog hears noise, it will stand up, look around, and then make a confused look. Then it'll doze off again.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep
from preset_actions import shake_head

my_dog = Pidog()
sleep(0.1)

def loop_around(amplitude=60, interval=0.5, speed=100):
```

(continues on next page)

(continued from previous page)

```
    my_dog.head_move([[amplitude,0,0]], immediately=True, speed=speed)
    my_dog.wait_all_done()
    sleep(interval)
    my_dog.head_move([[-amplitude,0,0]], immediately=True, speed=speed)
    my_dog.wait_all_done()
    sleep(interval)
    my_dog.head_move([[0,0,0]], immediately=True, speed=speed)
    my_dog.wait_all_done()

def is_sound():
    if my_dog.ears.isdetected():
        direction = my_dog.ears.read()
        if direction != 0:
            return True
        else:
            return False
    else:
        return False

def rest():
    my_dog.wait_all_done()
    my_dog.do_action('lie', speed=50)
    my_dog.wait_all_done()

    while True:
        # Sleeping
        my_dog.rgb_strip.set_mode('breath', 'pink', bps=0.3)
        my_dog.head_move([[0,0,-40]], immediately=True, speed=5)
        my_dog.do_action('doze_off', speed=92)
        # Cleanup sound detection
        sleep(1)
        is_sound()

        # keep sleeping
        while is_sound() is False:
            my_dog.do_action('doze_off', speed=92)
            sleep(0.2)

        # If heard anything, wake up
        # Set light to yellow and stand up
        my_dog.rgb_strip.set_mode('boom', 'yellow', bps=1)
        my_dog.body_stop()
        my_dog.do_action('stand', speed=90)
        my_dog.head_move([[0, 0, 0]], immediately=True, speed=80)
        my_dog.wait_all_done()
        # Look arround
        loop_around(60, 1, 60)
        sleep(0.5)
        # tilt head and being confused
        my_dog.speak('confused_3', volume=80)
        my_dog.do_action('tilting_head_left', speed=80)
        my_dog.wait_all_done()
```

(continues on next page)

(continued from previous page)

```
        sleep(1.2)
        my_dog.head_move([[0, 0, -10]], immediately=True, speed=80)
        my_dog.wait_all_done()
        sleep(0.4)
        # Shake head , mean to ignore it
        shake_head(my_dog)
        sleep(0.2)

        # Lay down again
        my_dog.rgb_strip.set_mode('breath', 'pink', bps=1)
        my_dog.do_action('lie', speed=50)
        my_dog.wait_all_done()
        sleep(1)


if __name__ == "__main__":
    try:
        rest()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        my_dog.close()
```

2.3.6 6. Be Picked Up

Try lifting your PiDog from the ground, PiDog will feel like it can fly, and it will cheer in a superman pose.

Run the Code

```
cd ~/pidog/examples
sudo python3 6_be_picked_up.py
```

After the program runs, the 6-DOF IMU Module will always calculate the acceleration in the vertical direction. If PiDog is calculated to be in a state of weightlessness, PiDog assumes a superman pose and cheers. Otherwise, consider PiDog to be on flat ground and make a standing pose.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep

my_dog = Pidog()
sleep(0.1)

```

(continues on next page)

(continued from previous page)

```
def fly():
    my_dog.rgb_strip.set_mode('boom', color='red', bps=3)
    my_dog.legs.servo_move([45, -45, 90, -80, 90, 90, -90, -90], speed=60)
    my_dog.do_action('wag_tail', step_count=10, speed=100)
    my_dog.speak('woohoo', volume=80)
    my_dog.wait_legs_done()
    sleep(1)

def stand():
    my_dog.rgb_strip.set_mode('breath', color='green', bps=1)
    my_dog.do_action('stand', speed=60)
    my_dog.wait_legs_done()
    sleep(1)

def be_picked_up():
    isUp = False
    upflag = False
    downflag = False

    stand()

    while True:
        ax = my_dog.accData[0]
        print('ax: %s, is up: %s' % (ax, isUp))

        # gravity : 1G = -16384
        if ax < -18000: # if down, acceleration is in the same direction as gravity, ax
↪< -1G
            my_dog.body_stop()
            if upflag == False:
                upflag = True
            if downflag == True:
                isUp = False
                downflag = False
                stand()

        if ax > -13000: # if up, acceleration is the opposite of gravity, ax will > -1G
            my_dog.body_stop()
            if upflag == True:
                isUp = True
                upflag = False
                fly()
            if downflag == False:
                downflag = True

        sleep(0.02)


if __name__ == "__main__":
    try:
        be_picked_up()
```

(continues on next page)

(continued from previous page)

```
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        my_dog.close()
```

2.3.7 7. Face Track

PiDog will sit quietly in place. You applaud it, it looks your way, and if it sees you, it says hello.

Run the Code

```
cd ~/pidog/examples
sudo python3 7_face_track.py
```

After running this code, PiDog will start the camera and enable the face detection function. You can visit `http:// + PiDog's IP +/mjpg` (like mine is `http://192.168.18.138:9000/mjpg`) in your browser to view the camera's picture.

Then PiDog will sit down and activate the Sound Direction Sensor Module to detect the direction of your clapping. When PiDog hears clapping (or other noise), it turns its head toward the sound source, trying to find you.

If it sees you (face detection finds an object), it will wag its tail and let out a bark.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep
from vilib import Vilib
from preset_actions import bark

my_dog = Pidog()
sleep(0.1)

def face_track():
    Vilib.camera_start(vflip=False, hflip=False)
    Vilib.display(local=True, web=True)
    Vilib.human_detect_switch(True)
    sleep(0.2)
    print('start')
    yaw = 0
    roll = 0
    pitch = 0
    flag = False
    direction = 0
```

(continues on next page)

(continued from previous page)

```
    my_dog.do_action('sit', speed=50)
    my_dog.head_move([[yaw, 0, pitch]], pitch_comp=-40, immediately=True, speed=80)
    my_dog.wait_all_done()
    sleep(0.5)
    # Cleanup sound detection by servos moving
    if my_dog.ears.isdetected():
        direction = my_dog.ears.read()

    while True:
        if flag == False:
            my_dog.rgb_strip.set_mode('breath', 'pink', bps=1)
        # If heard somthing, turn to face it
        if my_dog.ears.isdetected():
            flag = False
            direction = my_dog.ears.read()
            pitch = 0
            if direction > 0 and direction < 160:
                yaw = -direction
                if yaw < -80:
                    yaw = -80
            elif direction > 200 and direction < 360:
                yaw = 360 - direction
                if yaw > 80:
                    yaw = 80
            my_dog.head_move([[yaw, 0, pitch]], pitch_comp=-40, immediately=True,␣
↪speed=80)
            my_dog.wait_head_done()
            sleep(0.05)

        ex = Vilib.detect_obj_parameter['human_x'] - 320
        ey = Vilib.detect_obj_parameter['human_y'] - 240
        people = Vilib.detect_obj_parameter['human_n']

        # If see someone, bark at him/her
        if people > 0 and flag == False:
            flag = True
            my_dog.do_action('wag_tail', step_count=2, speed=100)
            bark(my_dog, [yaw, 0, 0], pitch_comp=-40, volume=80)
            if my_dog.ears.isdetected():
                direction = my_dog.ears.read()

        if ex > 15 and yaw > -80:
            yaw -= 0.5 * int(ex/30.0+0.5)

        elif ex < -15 and yaw < 80:
            yaw += 0.5 * int(-ex/30.0+0.5)

        if ey > 25:
            pitch -= 1*int(ey/50+0.5)
            if pitch < - 30:
                pitch = -30
        elif ey < -25:
```

(continues on next page)

(continued from previous page)

```
            pitch += 1*int(-ey/50+0.5)
            if pitch > 30:
                pitch = 30

        print('direction: %s |number: %s | ex, ey: %s, %s | yrp: %s, %s, %s '
            % (direction, people, ex, ey, round(yaw, 2), round(roll, 2), round(pitch,␣
↪2)),
            end='\r',
            flush=True,
            )
        my_dog.head_move([[yaw, 0, pitch]], pitch_comp=-40, immediately=True, speed=100)
        sleep(0.05)


if __name__ == "__main__":
    try:
        face_track()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        Vilib.camera_close()
        my_dog.close()
```

2.3.8 8. Push Up

PiDog is an exercise-loving robot that will do push-ups with you.

Run the Code

```
cd ~/pidog/examples
sudo python3 8_pushup.py
```

After the program runs, PiDog will perform a plank, then cycle through push-ups and barks.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep
from preset_actions import push_up, bark

my_dog = Pidog()

sleep(0.5)

```

(continues on next page)

(continued from previous page)

```python

def main():
    my_dog.legs_move([[45, -25, -45, 25, 80, 70, -80, -70]], speed=50)
    my_dog.head_move([[0, 0, -20]], speed=90)
    my_dog.wait_all_done()
    sleep(0.5)
    bark(my_dog, [0, 0, -20])
    sleep(0.1)
    bark(my_dog, [0, 0, -20])

    sleep(1)
    my_dog.rgb_strip.set_mode("speak", color="blue", bps=2)
    while True:
        push_up(my_dog, speed=92)
        bark(my_dog, [0, 0, -40])
        sleep(0.4)


if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        my_dog.close()
```

2.3.9 9. Howling

PiDog is not only a cute puppy, but also a mighty dog. Come hear it howl!

Run the Code

```
cd ~/pidog/examples
sudo python3 9_howling.py
```

After the program runs, PiDog will sit on the ground and howl.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```python
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep
from preset_actions import howling

my_dog = Pidog()
```

(continues on next page)

(continued from previous page)

```
sleep(0.5)


def main():
    my_dog.do_action('sit', speed=50)
    my_dog.head_move([[0, 0, 0]], pitch_comp=-40, immediately=True, speed=80)
    sleep(0.5)
    while True:
        howling(my_dog)


if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        my_dog.close()
```

2.3.10 10. Balance

Because PiDog is equipped with a 6-DOF IMU module, it has a great sense of balance.

In this example, you can make PiDog walk smoothly on the table, even if you lift one side of the table, PiDog will walk smoothly on the gentle slope.

Run the Code

```
cd ~/pidog/examples
sudo python3 10_balance.py
```

After the program is running, you will see a printed keyboard on the terminal. You can control PiDog to walk smoothly on the ramp by typing the below keys.

Keys	Function
W	Forward
E	Stand
A	Turn Left
S	Backward
D	Turn Right
R	Each press slightly lifts the body; multiple presses are needed for a noticeable rise.
F	Each press lowers the body a bit; it takes multiple presses for a noticeable descent.

Code

Please find the code in .

2.3.11 11. Play PiDog with Keyboard

In this example, we will use the keyboard to control PiDog. You can press these keys in the terminal to make it act.

Keys	Function	Keys	Function	Keys	Function
1	doze off	q	bark harder	a	turn left
2	push-up	w	forward	s	backward
3	howling	e	pant	d	turn right
4	twist body	r	wag tail	f	shake head
5	scratch	t	hake head	g	high five
u	head roll	U	head roll+	z	lie
i	head pitch	I	head pitch+	x	stand up
o	head roll	O	head roll+	c	sit
j	head yaw	J	head yaw+	v	stretch
k	head pitch	K	head pitch+	m	head reset
l	head yaw	L	head yaw+	W	trot

Run the Code

```
cd ~/pidog/examples
sudo python3 11_keyboard_control.py
```

After the program runs, you will see a printed keyboard on the terminal. Now you can control PiDog with keyboard in terminal.

Code

Please find the code in .

2.3.12 12. Play PiDog with APP

In this example, we will use SunFounder Controller APP to control PiDog.

You need to download the APP on your phone/tablet first, then connect to the hotspot sent by PiDog, and finally create your own remote control on SunFounder Controller to control PiDog.

Control Pidog with app

1. Install SunFounder Controller from **APP Store(iOS)** or **Google Play(Android)**.
2. Install `sunfounder-controller` module.

   The `robot-hat`, `vilib`, and `picar-x` modules need to be installed first, for details see: *5. Install All the Modules(Important)*.

   ```
   cd ~
   git clone https://github.com/sunfounder/sunfounder-controller.git
   cd ~/sunfounder-controller
   sudo python3 setup.py install
   ```

3. Run the Code.

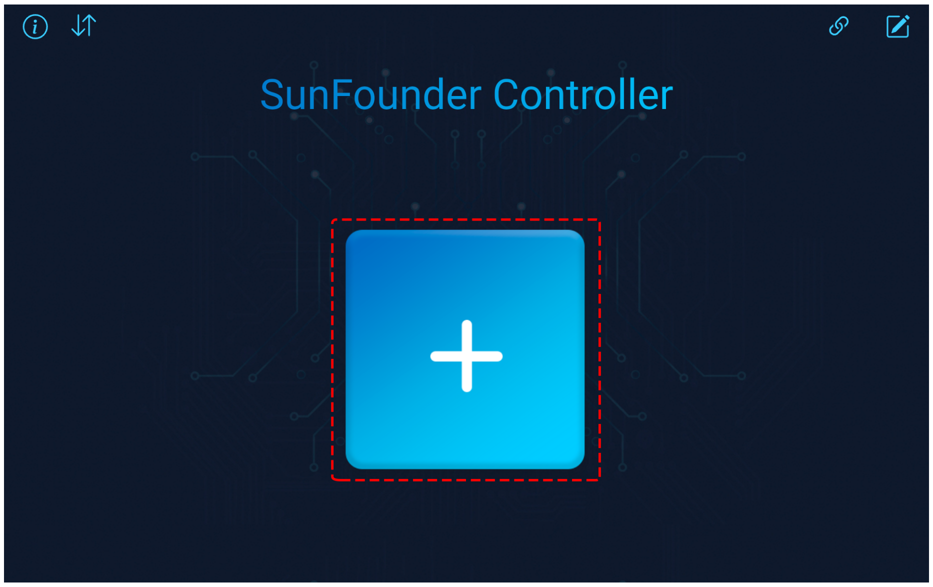
   ```
   cd ~/pidog/examples
   sudo python3 12_app_control.py
   ```

After the code runs, you will see the following prompt, which means your PiDog has successfully started network communication.

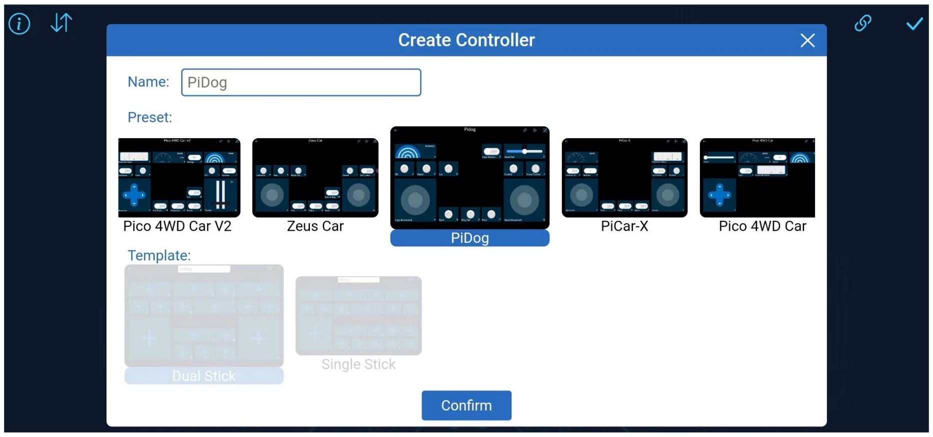
```
Running on: http://192.168.18.138:9000/mjpg

* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: development
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

4. Connect `PiDog` and `Sunfounder Controller`.

   - Connect your tablet/phone to the WLAN where PiDog is located.
   - Open the `Sunfounder Controller` APP. Click the + icon to add a controller.



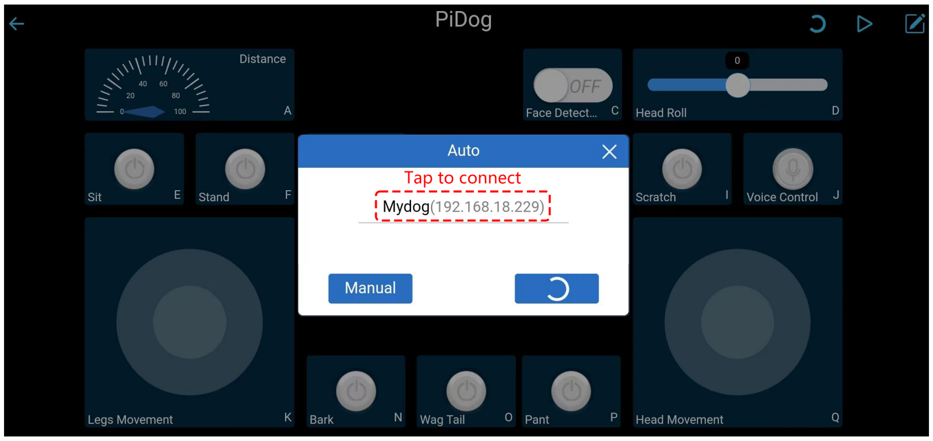
   - Preset controllers are available for some products, here we choose **PiDog**. Give it a name, or simply tap **Confirm**.


Create Controller

Name: PiDog

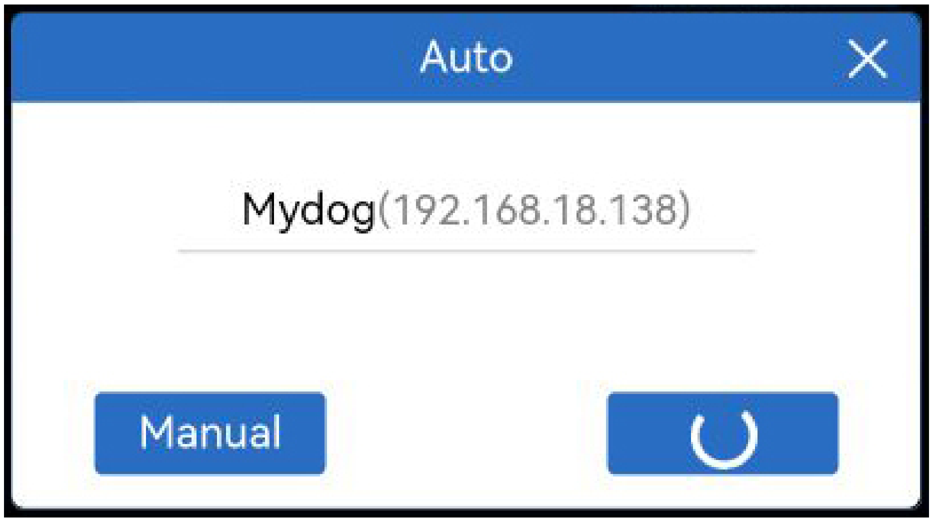
Preset:

   - Once inside, the app will automatically search for the **Mydog**. After a moment, you will see a prompt saying "Connected Successfully."



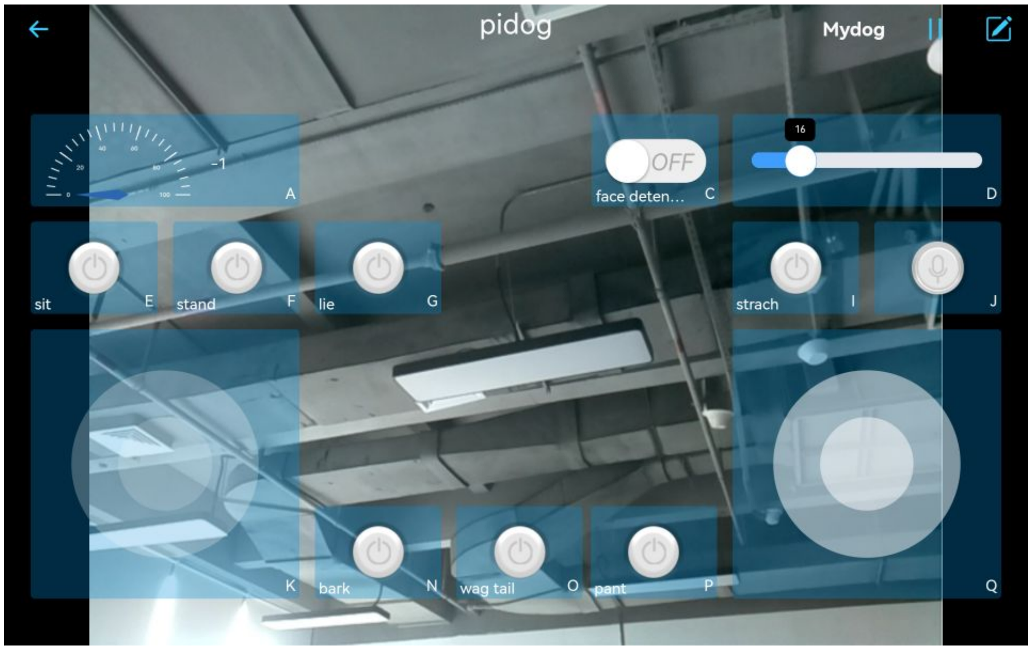
Note:

- You can also manually click the button. Wait a few seconds, MyDog(IP) will appear, click it to connect.



5. Run the Controller.

   - When the "Connected Successfully" prompt appears, tap the button in the upper right corner.
   - The picture taken by the camera will appear on the APP, and now you can control your PiDog with these widgets.



Here are the functions of the widgets.

- A: Detect the obstacle distance, that is, the reading of the ultrasonic module.
- C: Turn on/off face detection.
- D: Control PiDog's head tilt angle (tilt head).
- E: Sit.
- F: Stand.
- G: Lie.
- I: Scratch PiDog's head.
- N: Bark.
- O: Wag tail.
- P: Pant.
- K: Control PiDog's movement (forward, backward, left and right).
- Q: Controls the orientation of PiDog's head.
- J: Switch to voice control mode. It supports the following voice commands:
  - `forward`
  - `backward`
  - `turn left`
  - `turn right`
  - `trot`
  - `stop`
  - `lie down`
  - `stand up`

  - `sit`
  - `bark`
  - `bark harder`
  - `pant`
  - `wag tail`
  - `shake head`
  - `stretch`
  - `doze off`
  - `push-up`
  - `howling`
  - `twist body`
  - `scratch`
  - `handshake`
  - `high five`

Autostart on Boot

When controlling PiDog via the APP, you wouldn't want to first log into the Raspberry Pi and keep `12_app_control.py` running before connecting with the APP.

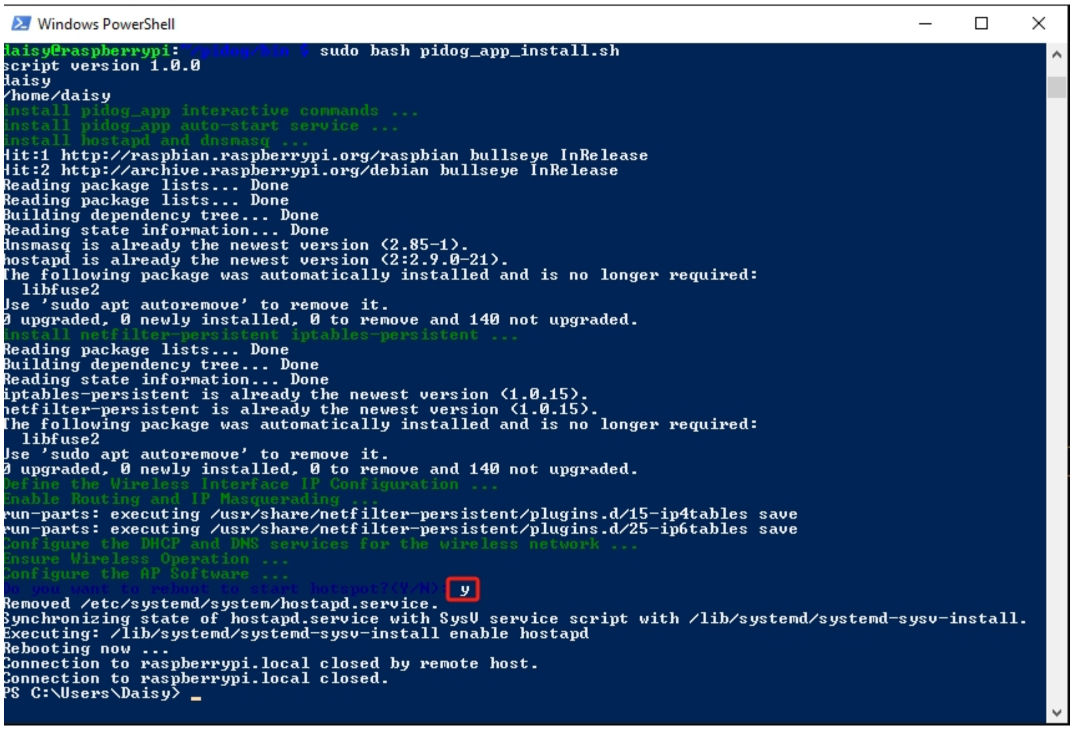
There's a more streamlined approach. You can set PiDog to automatically run `12_app_control.py` every time it's powered on. After this, you can directly connect to PiDog using the APP and control your robotic dog with ease.

How to set this up?

1. Execute the following commands to install and configure the `pidog_app` application and set up WiFi for PiDog.

   ```
   cd ~/pidog/bin
   sudo bash pidog_app_install.sh
   ```

2. At the end, input `y` to reboot PiDog.



```
daisy@raspberrypi:~/pidog/bin $ sudo bash pidog_app_install.sh
script version 1.0.0
daisy
/home/daisy
install pidog_app interactive commands ...
install pidog_app auto-start service ...
install hostapd and dnsmasq ...
Hit:1 http://raspbian.raspberrypi.org/raspbian bullseye InRelease
Hit:2 http://archive.raspberrypi.org/debian bullseye InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
dnsmasq is already the newest version (2.85-1).
hostapd is already the newest version (2:2.9.0-21).
The following package was automatically installed and is no longer required:
  libfuse2
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 140 not upgraded.
install netfilter-persistent iptables-persistent ...
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
iptables-persistent is already the newest version (1.0.15).
netfilter-persistent is already the newest version (1.0.15).
The following package was automatically installed and is no longer required:
  libfuse2
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 140 not upgraded.
Define the Wireless Interface IP Configuration ...
Enable Routing and IP Masquerading ...
run-parts: executing /usr/share/netfilter-persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
Configure the DHCP and DNS services for the wireless network ...
Ensure Wireless Operation ...
Configure the AP Software ...
Do you want to reboot to start hotspot?(Y/N): y
Removed /etc/systemd/system/hostapd.service.
Synchronizing state of hostapd.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable hostapd
Rebooting now ...
Connection to raspberrypi.local closed by remote host.
Connection to raspberrypi.local closed.
PS C:\Users\Daisy> _
```

3. From then on, you can simply power on PiDog and control it directly using the APP.

Warning: If you wish to run other scripts, first execute `pidog_app disable` to turn off the autostart feature.

APP Program Configuration

You can input the following commands to modify the APP mode's settings.

```
pidog_app <OPTION> [input]
```

OPTION

- `-h` `help`: help, show this message
- `start` `restart`: restart `pidog_app` service
- `stop`: stop `pidog_app` service
- `disable`: disable auto-start `app_controller` program on bootstrap
- `enable`: enable auto-start `app_controller` program on bootstrap
- `close_ap`: close hotspot, disable auto-start hotspot on boot and switch to sta mode
- `open_ap`: open hotspot, enable auto-start hotspot on boot
- `ssid`: set the ssid (network name) of the hotspot
- `psk`: set the password of the hotspot
- `country`: set the country code of the hotspot

2.3.13 13. Ball Track

PiDog will sit quietly in place. You put a red ball in front of it, it will stand, and then chase the ball.

Run the Code

```
cd ~/pidog/examples
sudo python3 13_ball_track.py
```

After running this code, PiDog will start the camera. You can visit `http://+ PiDog's IP +/mjpg` (like mine is `http://192.168.18.138:9000/mjpg`) in your browser to view the camera's picture.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep
from vilib import Vilib
from preset_actions import bark

my_dog = Pidog()

sleep(0.1)

STEP = 0.5

def delay(time):
    my_dog.wait_legs_done()
    my_dog.wait_head_done()
    sleep(time)

def ball_track():
    Vilib.camera_start(vflip=False, hflip=False)
    Vilib.display(local=True, web=True)
    Vilib.color_detect_switch(True)
    sleep(0.2)
    print('start')
    yaw = 0
    roll = 0
    pitch = 0
    flag = False
    direction = 0

    my_dog.do_action('stand', speed=50)
    my_dog.head_move([[yaw, 0, pitch]], immediately=True, speed=80)
    delay(0.5)

    while True:

        ball_x = Vilib.detect_obj_parameter['color_x'] - 320
```

(continues on next page)

(continued from previous page)

```
        ball_y = Vilib.detect_obj_parameter['color_y'] - 240
        width = Vilib.detect_obj_parameter['color_w']

        if ball_x > 15 and yaw > -80:
            yaw -= STEP

        elif ball_x < -15 and yaw < 80:
            yaw += STEP

        if ball_y > 25:
            pitch -= STEP
            if pitch < - 40:
                pitch = -40
        elif ball_y < -25:
            pitch += STEP
            if pitch > 20:
                pitch = 20

        print(f"yaw: {yaw}, pitch: {pitch}, width: {width}")

        my_dog.head_move([[yaw, 0, pitch]], immediately=True, speed=100)
        if width == 0:
            pitch = 0
            yaw = 0
        elif width < 300:
            if my_dog.is_legs_done():
                if yaw < -30:
                    print("turn right")
                    my_dog.do_action('turn_right', speed=98)
                elif yaw > 30:
                    print("turn left")
                    my_dog.do_action('turn_left', speed=98)
                else:
                    my_dog.do_action('forward', speed=98)
        sleep(0.02)


if __name__ == "__main__":
    try:
        ball_track()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        Vilib.camera_close()
        my_dog.close()
```

Then you may want to master its basic functions, or write some fun examples.

If you are familiar with Python programming, you can find examples of PiDog's basic functions in the `~/pidog/basic_examples` directory.

If you prefer, you can master them in a step-by-step fashion using the lessons provided below.

2.4 4. Easy Coding

Here, we delve into various functions, breaking them down for a comprehensive understanding. Each sub-topic is dedicated to a specific function, making it easier for you to grasp and implement them. Whether it's initiating parameters, controlling specific movements, or incorporating sensory inputs, we've covered them all. Navigate through the sub-topics below to kickstart your coding journey with Pidog.

2.4.1 1. PiDog Initialization

The functions of PiDog are written in the `Pidog` class, and the prototype of this class is shown below.

```
Class: Pidog()

__init__(leg_pins=DEFAULT_LEGS_PINS,
        head_pins=DEFAULT_HEAD_PINS,
        tail_pin=DEFAULT_TAIL_PIN,
        leg_init_angles=None,
        head_init_angles=None,
        tail_init_angle=None)
```

PiDog must be instantiated in one of several ways, as shown below.

1. Following are the simplest steps of initialization.

```
# Import Pidog class
from pidog import Pidog

# instantiate a Pidog
my_dog = Pidog()
```

2. PiDog has 12 servos, which can be initialized when we instantiate it.

```
# Import Pidog class
from pidog import Pidog

# instantiate a Pidog with custom initialized servo angles
my_dog = Pidog(leg_init_angles = [25, 25, -25, -25, 70, -45, -70, 45],
                head_init_angles = [0, 0, -25],
                tail_init_angle= [0]
            )
```

In the `Pidog` class, the servos are divided into three groups.

- `leg_init_angles` : In this array, 8 values determine the angles of eight servos, with the servos (pin numbers) they control being `2, 3, 7, 8, 0, 1, 10, 11`. From the foldout, you can see where these servos are located.
- `head_init_angles` : There is an array with 3 values, controllers for PiDog-head yaw, roll, pitch servos (`no. 4, 6, 5`) which react to yaw, roll, pitch, or Deflection of the body.
- `tail_init_angle` : In this array, there is only one value, which is dedicated to controlling the tail servo, which is 9.

3. `Pidog` allows you to redefine the serial number of the servos when instantiating the robot if your servo order is different.

```python
# Import Pidog class
from pidog import Pidog

# instantiate a Pidog with custom initialized pins & servo angles
my_dog = Pidog(leg_pins=[2, 3, 7, 8, 0, 1, 10, 11],
                head_pins=[4, 6, 5],
                tail_pin=[9],
                leg_init_angles = [25, 25, -25, -25, 70, -45, -70, 45],
                head_init_angles = [0, 0, -25],
                tail_init_angle= [0]
            )
```

2.4.2 2. Leg Move

PiDog's leg movements are implemented by the following functions.

```python
Pidog.legs_move(target_angles, immediately=True, speed=50)
```

- `target_angles`: It is a two-dimensional array composed of an array of 8 servo angles (referred to as angle group) as elements. These angle groups will be used to control the angles of the 8 foot servos. If multiple angle groups are written, the unexecuted angle groups will be stored in the cache.
- `immediately` : When calling the function, set this parameter to `True`, the cache will be cleared immediately to execute the newly written angle group; if the parameter is set to `False`, the newly written The incoming angular group is added to the execution queue.
- `speed` : The speed at which the angular group is executed.

Some common usages are listed below:

1. Take action immediately.

```python
from pidog import Pidog
import time

my_dog = Pidog()

# half stand
my_dog.legs_move([[45, 10, -45, -10, 45, 10, -45, -10]], speed=50)
```

2. Add some angular groups to the execution queue.

```python
from pidog import Pidog
import time

my_dog = Pidog()

# half stand
my_dog.legs_move([[45, 10, -45, -10, 45, 10, -45, -10]], speed=50)

# multiple actions
my_dog.legs_move([[45, 35, -45, -35, 80, 70, -80, -70],
                    [90, -30, -90, 30, 80, 70, -80, -70],
                    [45, 35, -45, -35, 80, 70, -80, -70]],  immediately=False, speed=30)
```

3. Perform repetitions within 10 seconds.

```
from pidog import Pidog
import time

my_dog = Pidog()

# half stand
my_dog.legs_move([[45, 10, -45, -10, 45, 10, -45, -10]], speed=50)

# pushup preparation
my_dog.legs_move([[45, 35, -45, -35, 80, 70, -80, -70]], immediately=False, speed=20)

# pushup
for _ in range(99):
    my_dog.legs_move([[90, -30, -90, 30, 80, 70, -80, -70],
                        [45, 35, -45, -35, 80, 70, -80, -70]],  immediately=False,␣
↪speed=30)

# keep 10s
time.sleep(10)

# stop and half stand
my_dog.legs_move([[45, 10, -45, -10, 45, 10, -45, -10]], immediately=True, speed=50)
```

PiDog's leg control also has the following functions that can be used together:

```
Pidog.is_legs_done()
```

This function is used to determine whether the angle group in the cache has been executed. If yes, return `True`; otherwise, return `False`.

```
Pidog.wait_legs_done()
```

Suspends the program until the angle groups in the cache have been executed.

```
Pidog.legs_stop()
```

Empty the angular group in the cache.

2.4.3 3. Head Move

The control of PiDog's head servo is implemented by the following functions.

```
Pidog.head_move(target_yrps, roll_comp=0, pitch_comp=0, immediately=True, speed=50)
```

- `target_angles` : It is a two-dimensional array composed of an array of 3 servo angles (referred to as angle group) as elements. These angle groups will be used to control the angles of the 8 foot servos. If multiple angle groups are written, the unexecuted angle groups will be stored in the cache.
- `roll_comp` : Provides angular compensation on the roll axis.
- `pitch_comp` : Provides angle compensation on the pitch axis.

- `immediately` : When calling the function, set this parameter to `True`, the cache will be cleared immediately to execute the newly written angle group; if the parameter is set to `False`, the newly written The incoming angular group is added to the execution queue.
- `speed` : The speed at which the angular group is executed.

PiDog's head servo control also has some supporting functions:

```
Pidog.is_head_done()
```

Whether all the head actions in the buffer to be executed

```
Pidog.wait_head_done()
```

Wait for all the head actions in the buffer to be executed

```
Pidog.head_stop()
```

Clear all the head actions of leg in the buffer, to make head servos stop

Here are some common use cases:

1. Nod five times.

```
from pidog import Pidog
import time

my_dog = Pidog()

for _ in range(5):
    my_dog.head_move([[0, 0, 30],[0, 0, -30]], speed=80)
    my_dog.wait_head_done()
    time.sleep(0.5)
```

2. Shake your head for 10 seconds.

```
from pidog import Pidog
import time

my_dog = Pidog()

for _ in range(99):
    my_dog.head_move([[30, 0, 0],[-30, 0, 0]], immediately=False, speed=30)

# keep 10s
time.sleep(10)

my_dog.head_move([[0, 0, 0]], immediately=True, speed=80)
```

3. Whether sitting or half standing, PiDog keeps its head level when shaking its head.

```
from pidog import Pidog
import time

my_dog = Pidog()
```

(continues on next page)

(continued from previous page)

```
# action list
shake_head = [[30, 0, 0],[-30, 0, 0]]
half_stand_leg = [[45, 10, -45, -10, 45, 10, -45, -10]]
sit_leg = [[30, 60, -30, -60, 80, -45, -80, 45]]

while True:
    # shake head in half stand
    my_dog.legs_move(half_stand_leg, speed=30)
    for _ in range(5):
        my_dog.head_move(shake_head, pitch_comp=0, speed=50)
    my_dog.wait_head_done()
    time.sleep(0.5)

    # shake head in sit
    my_dog.legs_move(sit_leg, speed=30)
    for _ in range(5):
        my_dog.head_move(shake_head, pitch_comp=-30, speed=50)
    my_dog.wait_head_done()
    time.sleep(0.5)
```

2.4.4 4. Tail Move

Following are the functions that control PiDog's tail. This function is similar to *2. Leg Move*.

```
Pidog.tail_move(target_angles, immediately=True, speed=50)
```

- `target_angles` : It is a two-dimensional array composed of an array of 1 servo angles (referred to as angle group) as elements. These angle groups will be used to control the angles of the 8 foot servos. If multiple angle groups are written, the unexecuted angle groups will be stored in the cache.
- `immediately` : When calling the function, set this parameter to `True`, the cache will be cleared immediately to execute the newly written angle group; if the parameter is set to `False`, the newly written The incoming angular group is added to the execution queue.
- `speed` : The speed at which the angular group is executed.

PiDog's tail servo control also has some supporting functions:

```
Pidog.is_tail_done()
```

whether all the tail actions in the buffer to be executed

```
Pidog.wait_tail_done()
```

wait for all the tail actions in the buffer to be executed

```
Pidog.tail_stop()
```

clear all the tail actions of leg in the buffer, to make tail servo stop

Here are some common usages:

1. Wag tail for 10 seconds.

```
from pidog import Pidog
import time

my_dog = Pidog()

for _ in range(99):
    my_dog.tail_move([[30],[-30]], immediately=False, speed=30)

# keep 10s
time.sleep(10)

my_dog.tail_stop()
```

2.4.5 5. Stop All Actions

After the previous chapters, you can find that the servo control of PiDog is divided into three threads. This allows PiDog's head and body to move at the same time, even with two lines of code.

Here are a few functions that work with the three servo threads:

```
Pidog.wait_all_done()
```

Wait for all the actions in the leg actions buffer, head buffer and tail buffer to be executed

```
Pidog.body_stop()
```

Stop all the actions of legs, head and tail

```
Pidog.stop_and_lie()
```

Stop all the actions of legs, head and tail, then reset to "lie" pose

```
Pidog.close()
```

Stop all the actions, reset to "lie" pose, and close all the threads, usually used when exiting a program

Here are some common usages:

```
from pidog import Pidog
import time

my_dog = Pidog()

try:
    # pushup prepare
    my_dog.legs_move([[45, 35, -45, -35, 80, 70, -80, -70]], speed=30)
    my_dog.head_move([[0, 0, 0]], pitch_comp=-10, speed=80)
    my_dog.wait_all_done() # wait all the actions to be done
    time.sleep(0.5)

    # pushup
    leg_pushup_action = [
        [90, -30, -90, 30, 80, 70, -80, -70],
```

(continues on next page)

(continued from previous page)

```
        [45, 35, -45, -35, 80, 70, -80, -70],
    ]
    head_pushup_action = [
        [0, 0, -30],
        [0, 0, 20],
    ]

    # fill action buffers
    for _ in range(50):
        my_dog.legs_move(leg_pushup_action, immediately=False, speed=50)
        my_dog.head_move(head_pushup_action, pitch_comp=-10, immediately=False, speed=50)

    # show buffer length
    print(f"legs buffer length (start): {len(my_dog.legs_action_buffer)}")

    # keep 5 second & show buffer length
    time.sleep(5)
    print(f"legs buffer length (5s): {len(my_dog.legs_action_buffer)}")

    # stop action & show buffer length
    my_dog.stop_and_lie()
    print(f"legs buffer length (stop): {len(my_dog.legs_action_buffer)}")

except KeyboardInterrupt:
    pass
except Exception as e:
    print(f"\033[31mERROR: {e}\033[m")
finally:
    print("closing ...")
    my_dog.close() # close all the servo threads
```

2.4.6 6. Do Preset Action

Some commonly used actions have been pre-written in PiDog's library. You can call the following function to make PiDog do these actions directly.

```
Pidog.do_action(action_name, step_count=1, speed=50)
```

- `action_name` : Action name, the following strings can be written.
  - "sit"
  - "half_sit"
  - "stand"
  - "lie"
  - "lie_with_hands_out"
  - "forward"
  - "backward"
  - "turn_left"

  - "turn_right"
  - "trot"
  - "stretch"
  - "pushup"
  - "doze_off"
  - "nod_lethargy"
  - "shake_head"
  - "tilting_head_left"
  - "tilting_head_right"
  - "tilting_head"
  - "head_bark"
  - "head_up_down"
  - "wag_tail"

- step_count : How many times to perform this action.
- speed : How fast to perform the action.

Here is an example of usage:

1. Do ten push-ups, then sit on the floor and act cute.

```
from pidog import Pidog
import time

my_dog = Pidog()

try:
    # pushup
    my_dog.do_action("half_sit", speed=60)
    my_dog.do_action("pushup", step_count=10, speed=60)
    my_dog.wait_all_done()

    # act cute
    my_dog.do_action("sit", speed=60)
    my_dog.do_action("wag_tail", step_count=100,speed=90)
    my_dog.do_action("tilting_head", step_count=5, speed=20)
    my_dog.wait_head_done()

    my_dog.stop_and_lie()

except KeyboardInterrupt:
    pass
except Exception as e:
    print(f"\033[31mERROR: {e}\033[m")
finally:
    print("closing ...")
    my_dog.close()
```

2.4.7 7. PiDog Speak

PiDog can make sound, it is actually playing a piece of audio.

These audios are saved under `pidog\sounds` path, you can call the following function to play them.

```
Pidog.speak(name)
```

- `name` : Filename (without suffix), such as `"angry"`. `Pidog` provides the following audio.
  - `"angry"`
  - `"confused_1"`
  - `"confused_2"`
  - `"confused_3"`
  - `"growl_1"`
  - `"growl_2"`
  - `"howling"`
  - `"pant"`
  - `"single_bark_1"`
  - `"single_bark_2"`
  - `"snoring"`
  - `"woohoo"`

Here is an example of usage:

```
# !/usr/bin/env python3
''' play sound effecfs
    Note that you need to run with "sudo"
API:
    Pidog.speak(name, volume=100)
        play sound effecf in the file "../sounds"
        - name    str, file name of sound effect, no suffix required, eg: "angry"
        - volume  int, volume 0-100, default 100
'''
from pidog import Pidog
import os
import time

# change working directory
abspath = os.path.abspath(os.path.dirname(__file__))
# print(abspath)
os.chdir(abspath)

my_dog = Pidog()

print("\033[033mNote that you need to run with \"sudo\", otherwise there may be no sound.
↪\033[m")

# my_dog.speak("angry")
```

(continues on next page)

(continued from previous page)

```
# time.sleep(2)

for name in os.listdir('../sounds'):
    name = name.split('.')[0] # remove suffix
    print(name)
    my_dog.speak(name)
    # my_dog.speak(name, volume=50)
    time.sleep(3) # Note that the duration of each sound effect is different
print("closing ...")
my_dog.close()
```

2.4.8 8. Read Distance

Through the Ultrasonic Module in its head, PiDog can detect obstacles ahead.

An ultrasonic module can detect objects between 2 and 400 cm away.

With the following function, you can read the distance as a floating point number.

```
Pidog.ultrasonic.read_distance()
```

Here is an example of usage:

```
from pidog import Pidog
import time

my_dog = Pidog()
while True:
    distance = my_dog.ultrasonic.read_distance()
    distance = round(distance,2)
    print(f"Distance: {distance} cm")
    time.sleep(0.5)
```

2.4.9 9. PiDog RGB Strip

There is an RGB Strip on PiDog's chest, which PiDog can use to express emotions.

You can call the following function to control it.

```
Pidog.rgb_strip.set_mode(style='breath', color='white', bps=1, brightness=1):
```

- `style` : The lighting display mode of RGB Strip, the following are its available values.
  - `breath`
  - `boom`
  - `bark`
- `color` : The lights of the RGB Strip show the colors. You can enter 16-bit RGB values, such as `#a10a0a`, or the following color names.
  - `"white"`
  - `"black"`

  - "white"
  - "red"
  - "yellow"
  - "green"
  - "blue"
  - "cyan"
  - "magenta"
  - "pink"

- `brightness` : RGB Strip lights display brightness, you can enter a floating-point value from 0 to 1, such as `0.5`.
- `delay` : Float, display animation speed, the smaller the value, the faster the change.

Use the following statement to disable RGB Striping.

```
Pidog.rgb_strip.close()
```

Here are examples of their use:

```
from pidog import Pidog
import time

my_dog = Pidog()

while True:
    # style="breath", color="pink"
    my_dog.rgb_strip.set_mode(style="breath", color='pink')
    time.sleep(3)

    # style:"boom", color="#a10a0a"
    my_dog.rgb_strip.set_mode(style="bark", color="#a10a0a")
    time.sleep(3)

    # style:"boom", color="#a10a0a", brightness=0.5, bps=2.5
    my_dog.rgb_strip.set_mode(style="boom", color="#a10a0a", bps=2.5, brightness=0.5)
    time.sleep(3)

    # close
    my_dog.rgb_strip.close()
    time.sleep(2)
```

2.4.10 10. IMU Read

Through the 6-DOF IMU Module, PiDog can determine if it's standing on a slope, or if it's being picked up.

The 6-DOF IMU Module is equipped with a 3-axis accelerometer and a 3-axis gyroscope, allowing acceleration and angular velocity to be measured in three directions.

Note: Before using the module, make sure that it is correctly assembled. The label on the module will let you know if it is reversed.

You can read their acceleration with:

```
ax, ay, az = Pidog.accData
```

With the PiDog placed horizontally, the acceleration on the x-axis (ie ax) should be close to the acceleration of gravity (1g), with a value of -16384. The values of the y-axis and x-axis are close to 0.

Use the following way to read their angular velocity:

```
gx, gy, gz = my_dog.gyroData
```

In the case where PiDog is placed horizontally, all three values are close to 0.

Here are some examples of how 6-DOF Module is used:

1. Read real-time acceleration, angular velocity

```
from pidog import Pidog
import time

my_dog = Pidog()

my_dog.do_action("pushup", step_count=10, speed=20)

while True:
    ax, ay, az = my_dog.accData
    gx, gy, gz = my_dog.gyroData
    print(f"accData: {ax/16384:.2f} g ,{ay/16384:.2f} g, {az/16384:.2f} g       ␣
↪gyroData: {gx} °/s, {gy} °/s, {gz} °/s")
    time.sleep(0.2)
    if my_dog.is_legs_done():
        break

my_dog.stop_and_lie()

my_dog.close()
```

2. Calculate the lean angle of PiDog's body.

```
from pidog import Pidog
import time
import math

my_dog = Pidog()

while True:
    ax, ay, az = my_dog.accData
    body_pitch = math.atan2(ay,ax)/math.pi*180%360-180
    print(f"Body Degree: {body_pitch:.2f} °" )
    time.sleep(0.2)

my_dog.close()
```

3. While leaning, PiDog keeps its eyes level.

```python
from pidog import Pidog
import time
import math

my_dog = Pidog()

while True:
    ax, ay, az = my_dog.accData
    body_pitch = math.atan2(ay,ax)/math.pi*180%360-180
    my_dog.head_move([[0, 0, 0]], pitch_comp=-body_pitch, speed=80)
    time.sleep(0.2)

my_dog.close()
```

2.4.11 11. Sound Direction Detect

The PiDog has a Sound Direction Sensor Module that detects where sound is coming from, and we can trigger it by clapping near it.

Using this module is as simple as calling these functions.

```python
Pidog.ears.isdetected()
```

Returns `True` if sound is detected, `False` otherwise.

```python
Pidog.ears.read()
```

This function returns the direction of the sound source, with a range of 0 to 359; if the sound comes from the front, it returns 0; if it comes from the right, it returns 90.

An example of how to use this module is as follows:

```python
from pidog import Pidog

my_dog = Pidog()

while True:
    if my_dog.ears.isdetected():
        direction = my_dog.ears.read()
        print(f"sound direction: {direction}")
```

2.4.12 12. Pat the PiDog's Head

The Touch Swich on the head of PiDog can detect how you touch it. You can call the following functions to use it.

```python
Pidog.dual_touch.read()
```

- Touch the module from left to right (front to back for PiDog's orientation), it will return `"LS"`.
- Touch the module from right to left, it will return `"RS"`.
- Touch the module If the left side of the module is touched, it will return `"L"`.
- If the right side of the module is touched, it will return `"R"`.

- If the module is not touched, it will return "N".

Here is an example of its use:

```python
from pidog import Pidog
import time

my_dog = Pidog()
while True:
    touch_status = my_dog.dual_touch.read()
    print(f"touch_status: {touch_status}")
    time.sleep(0.5)
```

2.4.13 13. More

The following address will explain the use of PiDog's more basic functions:

- Vilib Library

  Vilib is a library developed by SunFounder for Raspberry Pi camera.

  It contains some practical functions, such as taking pictures, video recording, pose detection, face detection, motion detection, image classification and so on.

- SunFounder Controller

  SunFounder Controller is an application that allows users to customize the controller for controlling their robot or as an IoT platform.

CHAPTER
THREE

HARDWARE

When you are writing code, you may need to know how each module works or the role of each pin, then please see this chapter.

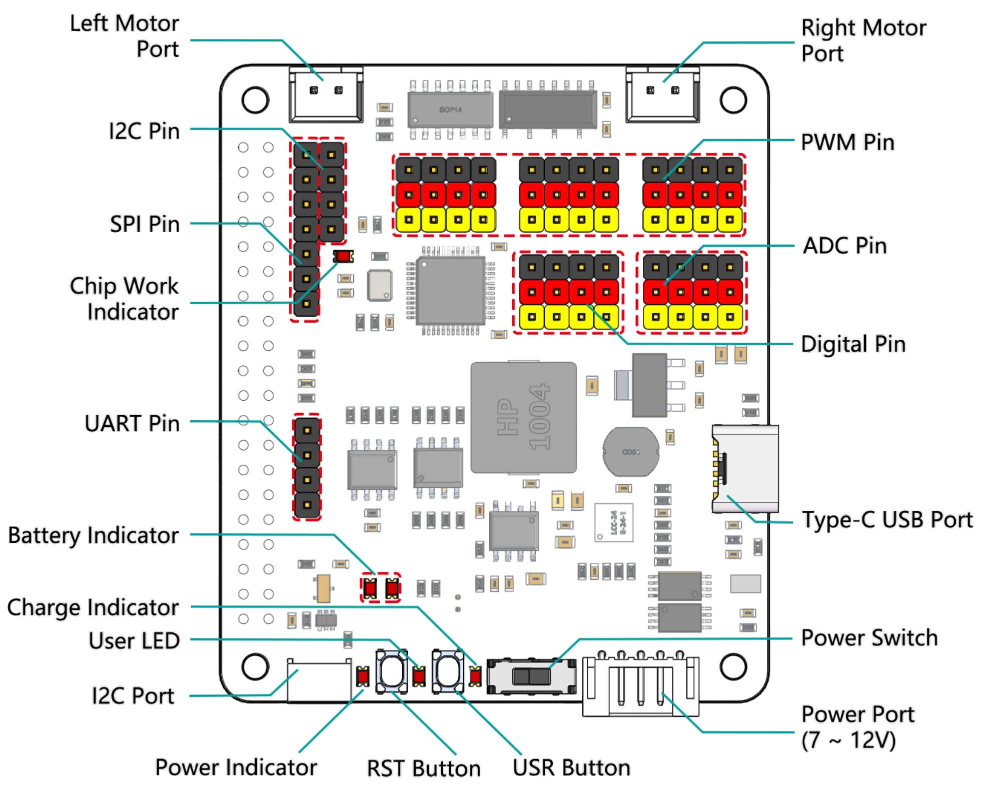
In this chapter you will find a description of each module's function, technical parameters and working principle.

3.1 Robot HAT

is a multifunctional expansion board that allows Raspberry Pi to be quickly turned into a robot. An MCU is on board to extend the PWM output and ADC input for the Raspberry Pi, as well as a motor driver chip, I2S audio module and mono speaker. As well as the GPIOs that lead out of the Raspberry Pi itself.

It also comes with a Speaker, which can be used to play background music, sound effects and implement TTS functions to make your project more interesting.

Accepts 7-12V PH2.0 5pin power input with 2 battery indicators, 1 charge indicator and 1 power indicator. The board also has a user available LED and a button for you to quickly test some effects.



Power Port

- 7-12V PH2.0 3pin power input.
- Powering the Raspberry Pi and Robot HAT at the same time.

Power Switch

- Turn on/off the power of the robot HAT.
- When you connect power to the power port, the Raspberry Pi will boot up. However, you will need to switch the power switch to ON to enable Robot HAT.

Type-C USB Port

- Insert the Type-C cable to charge the battery.
- At the same time, the charging indicator lights up in red color.
- When the battery is fully charged, the charging indicator turns off.
- If the USB cable is still plugged in about 4 hours after it is fully charged, the charging indicator will blink to prompt.

Digital Pin

- 4-channel digital pins, D0-D3.

ADC Pin

- 4-channel ADC pins, A0-A3.

PWM Pin

- 12-channel PWM pins, P0-P11.

Left/Right Motor Port

- 2-channel XH2.54 motor ports.
- The left port is connected to GPIO 4 and the right port is connected to GPIO 5.

I2C Pin and I2C Port

- **I2C Pin**: P2.54 4-pin interface.
- **I2C Port**: SH1.0 4-pin interface, which is compatible with QWIIC and STEMMA QT.
- These I2C interfaces are connected to the Raspberry Pi's I2C interface via GPIO2 (SDA) and GPIO3 (SCL).

SPI Pin

- P2.54 7-pin SPI interface.

UART Pin

- P2.54 4-pin interface.

RST Button

- The RST button, when using Ezblock, serves as a button to restart the Ezblock program.
- If not using Ezblock, the RST button does not have a predefined function and can be fully customized according to your needs.

USR Button

- The functions of USR Button can be set by your programming. (Pressing down leads to a input "0"; releasing produces a input "1".)

Battery Indicator

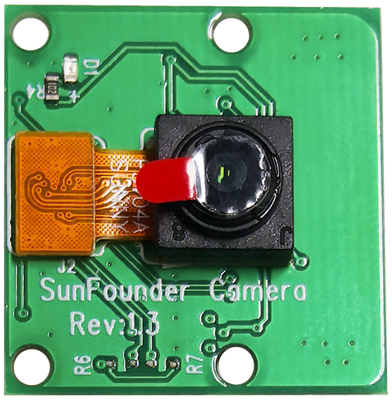
- Two LEDs light up when the voltage is higher than 7.6V.
- One LED lights up in the 7.15V to 7.6V range.
- Below 7.15V, both LEDs turn off.

Speaker and Speaker Port

- **Speaker**: This is a 2030 audio chamber speaker.
- **Speaker Port**: The Robot HAT is equipped with onboard I2S audio output, along with a 2030 audio chamber speaker, providing a mono sound output.

3.2 Camera Module

Description



This is a 5MP Raspberry Pi camera module with OV5647 sensor. It's plug and play, connect the included ribbon cable to the CSI (Camera Serial Interface) port on your Raspberry Pi and you're ready to go.

The board is small, about 25mm x 23mm x 9mm, and weighs 3g, making it ideal for mobile or other size and weight-critical applications. The camera module has a native resolution of 5 megapixels and has an on-board fixed focus lens that captures still images at 2592 x 1944 pixels, and also supports 1080p30, 720p60 and 640x480p90 video.

Note: The module is only capable of capturing pictures and videos, not sound.

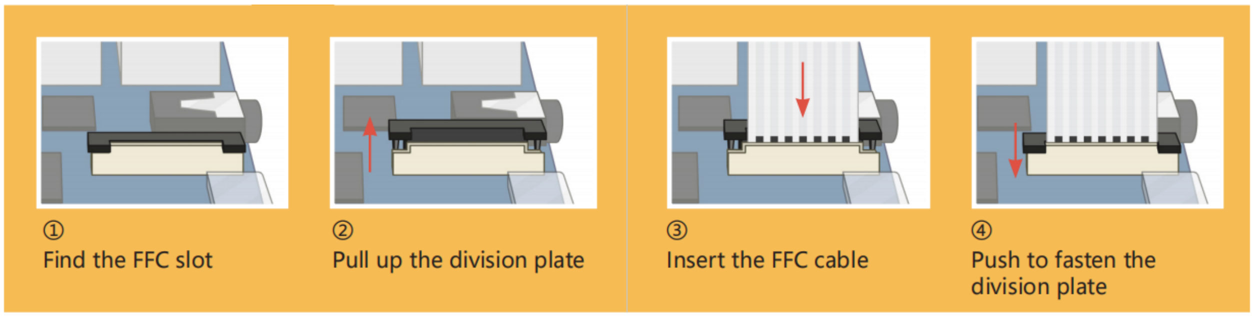
Specification

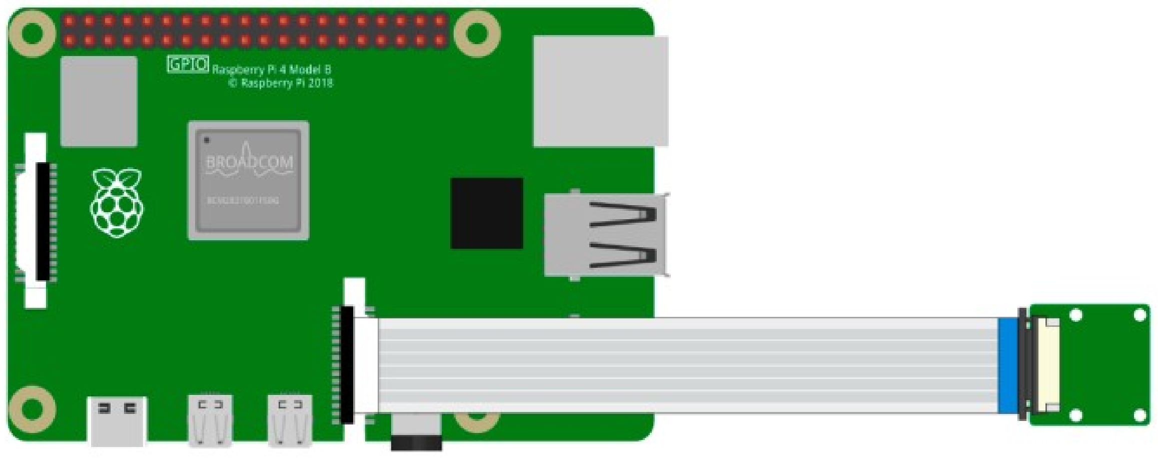
- **Static Images Resolution**: 2592×1944
- **Supported Video Resolution**: 1080p/30 fps, 720p/ 60fps and 640 x480p 60/90 video recording
- **Aperture (F)**: 1.8
- **Visual Angle**: 65 degree
- **Dimension**: 24mmx23.5mmx8mm
- **Weight**: 3g
- **Interface**: CSI connector
- **Supported OS**: Raspberry Pi OS(latest version recommended)

Assemble the Camera Module

On the camera module or Raspberry Pi, you will find a flat plastic connector. Carefully pull out the black fixing switch until the fixing switch is partially pulled out. Insert the FFC cable into the plastic connector in the direction shown and push the fixing switch back into place.

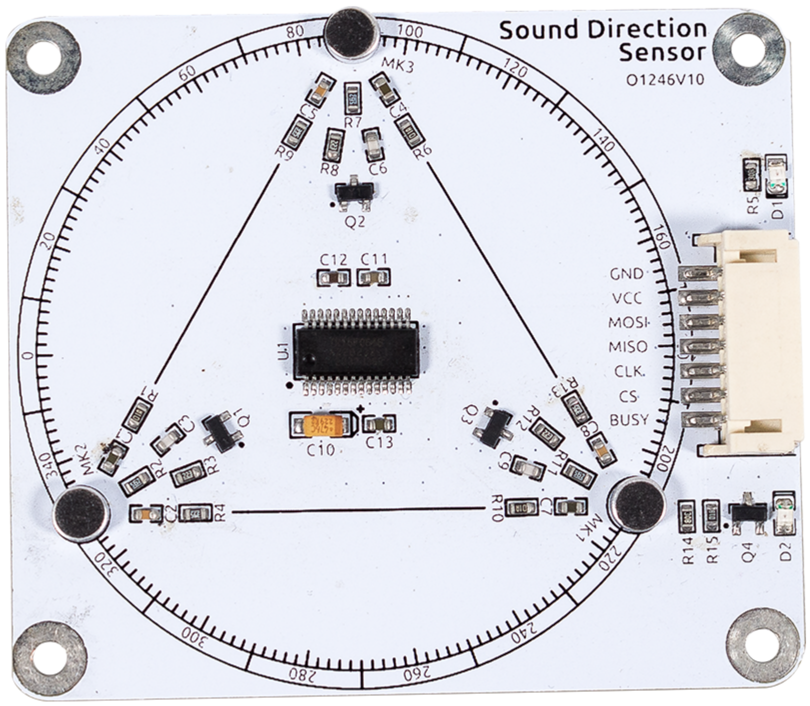
If the FFC wire is installed correctly, it will be straight and will not pull out when you gently pull on it. If not, reinstall it again.


① Find the FFC slot
② Pull up the division plate
③ Insert the FFC cable
④ Push to fasten the division plate



Warning: Do not install the camera with the power on, it may damage your camera.

3.3 Sound Direction Sensor



This is a sound direction recognition module. It is equipped with 3 microphones, which can detect sound sources from all directions, and is equipped with a TR16F064B, which is used to process sound signals and calculate the sound source direction. The minimum reconnaissance unit of this module is 20 degrees, and the data range is 0~360.

Data transmission process: the main controller pulls up the BUSY pin, and TR16F064B starts to monitor the direction. When 064B recognizes the direction, it will pull down the BUSY pin; When the main control detects that BUSY is low, it will send 16bit arbitrary data to 064B (follow the MSB transmission), and accept 16bit data, which is the sound direction data processed by 064B. After completion, the main control will pull the BUSY pin high to detect the direction again.

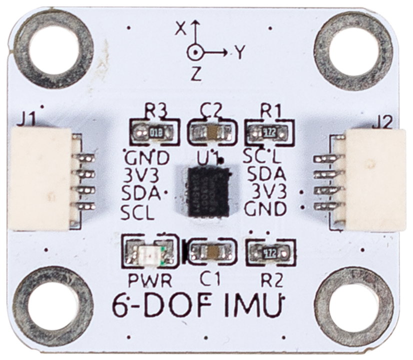
Specifications

- Power supply: 3.3V
- Communication: SPI
- Connector: PH2.0 7P
- Sound recognition angle range 360°
- Voice recognition angular accuracy ~10°

Pin Out

- GND - Ground Input
- VCC - 3.3V Power Supply Input
- MOSI - SPI MOSI
- MISO - SPI MISO
- SCLK - SPI clock
- CS - SPI Chip Select
- BUSY - busy detection

3.4 6-DOF IMU



The 6-DOF IMU is based on the SH3001.

SH3001 is a six-axis IMU (Inertial measurement unit). It integrates a three-axis gyroscope and a three-axis accelerometer. It is small in size and low in power consumption. It is suitable for consumer electronics market applications and can provide high-precision real-time angular velocity and linear acceleration data. The SH3001 has excellent temperature stability and can maintain high resolution within the operating range of -40°C to 85°C.

It is typically used in smartphones, tablet computers, multi-rotor drones, smart sweepers, page-turning laser pointers, AR/VR, smart remote controls, smart bracelets and other products.

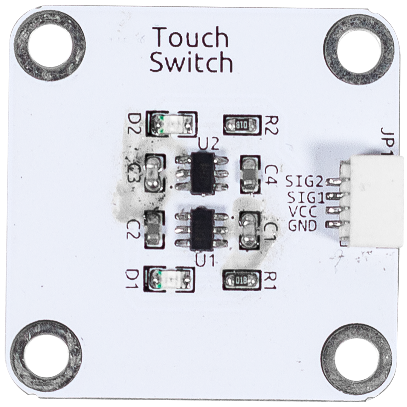
Specifications

- Power Supply: 3.3V
- Communication: IIC
- Connector: SH1.0 4P

Pin Out

- GND - Ground Input
- VCC - Power Supply Input
- SDA - IIC SDA
- SCL - IIC SCL

3.5 Dual Touch Sensor



Dual channel touch sensor, based on two ttp223 touch sensors. When a touch signal is detected, the corresponding pin level will be pulled low.

TTP223 is a touch pad detector IC that provides 1 touch key. The touch detection IC is specially designed to replace the traditional direct keys with different pad sizes. It features low power consumption and wide operating voltage.

Specifications

- Power Supply: 2.0V~5.5V
- Signal Output: Digital signal
- Connector: SH1.0 4P

Pin Out

- GND - Ground Input
- VCC - Power Supply Input
- SIG1 - Touch signal 1, low level means touch
- SIG2 - Touch signal 2, low level means touch

3.6 11-channel Light Board

This is an 11-channel RGB LED module, which is equipped with 11 RGB LEDs controlled by the SLED1735 chip.

SLED1734 can drive up to 256 LEDs and 75 RGB LEDs. In the LED matrix controlled by SLED1734, each LED has on/off, blinking, breathing light and automatic synchronization and many other functions. The chip has built-in PWM (pulse width modulation) technology, which can provide 256 levels of brightness adjustment. It also has a 16-level dot correction function.

Specifications

- Power supply: 3.3V
- Communication: IIC
- Connector: SH1.0 4P
- LEDs: 3535 RGB LEDs

Pin Out

- GND - Ground Input
- VCC - Power Supply Input
- SDA - IIC SDA
- SCL - IIC SCL

3.7 Ultrasonic Module

- **TRIG**: Trigger Pulse Input
- **ECHO**: Echo Pulse Output
- **GND**: Ground
- **VCC**: 5V Supply

This is the HC-SR04 ultrasonic distance sensor, providing non-contact measurement from 2 cm to 400 cm with a range accuracy of up to 3 mm. Included on the module is an ultrasonic transmitter, a receiver and a control circuit.

You only need to connect 4 pins: VCC (power), Trig (trigger), Echo (receive) and GND (ground) to make it easy to use for your measurement projects.

Features

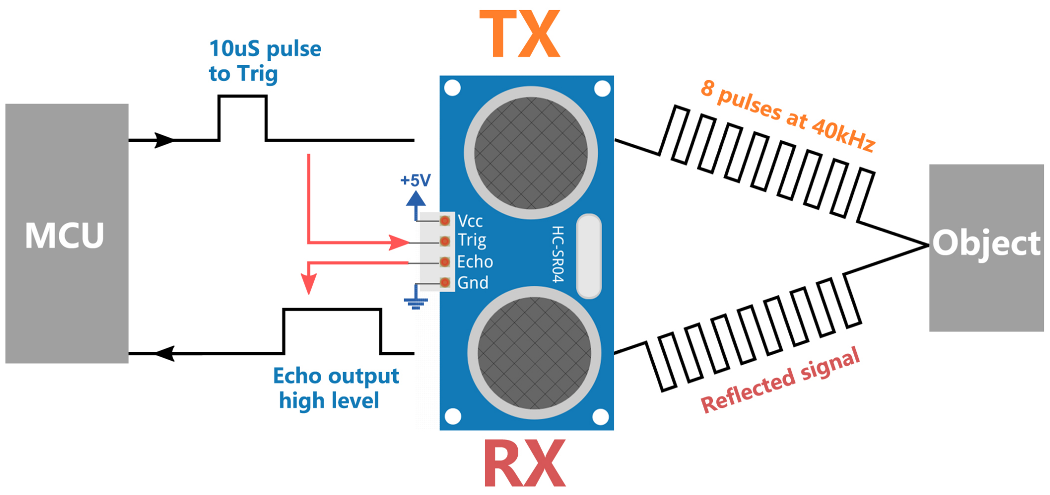
- Working Voltage: DC5V
- Working Current: 16mA
- Working Frequency: 40Hz
- Max Range: 500cm
- Min Range: 2cm
- Trigger Input Signal: 10uS TTL pulse
- Echo Output Signal: Input TTL lever signal and the range in proportion
- Connector: XH2.54-4P
- Dimension: 46x20.5x15 mm

Principle

The basic principles are as follows:

- Using IO trigger for at least 10us high level signal.
- The module sends an 8 cycle burst of ultrasound at 40 kHz and detects whether a pulse signal is received.

- Echo will output a high level if a signal is returned; the duration of the high level is the time from emission to return.
- Distance = (high level time x velocity of sound (340M/S)) / 2



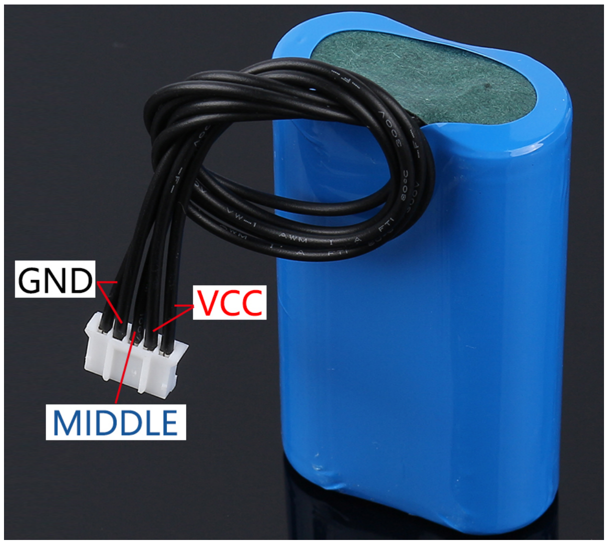
Formula:

- us / 58 = centimeters distance
- us / 148 = inch distance
- distance = high level time x velocity (340M/S) / 2

Application Notes

- This module should not be connected under power up, if necessary, let the module's GND be connected first. Otherwise, it will affect the work of the module.
- The area of the object to be measured should be at least 0.5 square meters and as flat as possible. Otherwise, it will affect results.

3.8 18650 Battery



- **VCC**: Battery positive terminal, here there are two sets of VCC and GND is to increase the current and reduce the resistance.
- **Middle**: To balance the voltage between the two cells and thus protect the battery.
- **GND**: Negative battery terminal.

This is a custom battery pack made by SunFounder consisting of two 18650 batteries with a capacity of 2000mAh. The connector is PH2.0-5P, which can be charged directly after being inserted into the Robot HAT.

Features

- Battery charge: 5V/2A
- Battery output: 5V/3A
- Battery capacity: 3.7V 2000mAh x 2
- Battery life: <90min
- Battery charge time: >130min
- Connector: PH2.0, 5P

CHAPTER

FOUR

APPENDIX

4.1 Filezilla Software

The File Transfer Protocol (FTP) is a standard communication protocol used for the transfer of computer files from a server to a client on a computer network.

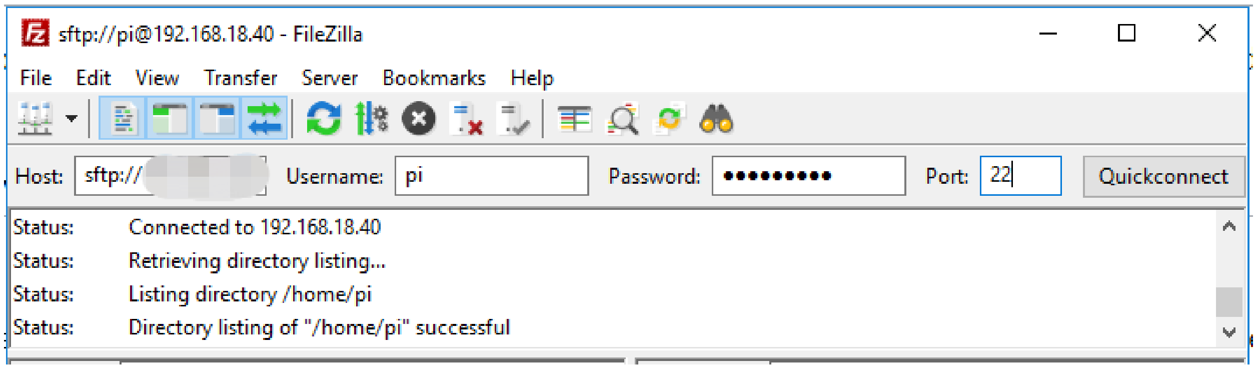
Filezilla is an open source software that not only supports FTP, but also FTP over TLS (FTPS) and SFTP. We can use Filezilla to upload local files (such as pictures and audio, etc.) to the Raspberry Pi, or download files from the Raspberry Pi to the local.

Step 1: Download Filezilla.

Download the client from Filezilla's official website, Filezilla has a very good tutorial, please refer to: Documentation - Filezilla.

Step 2: Connect to Raspberry Pi

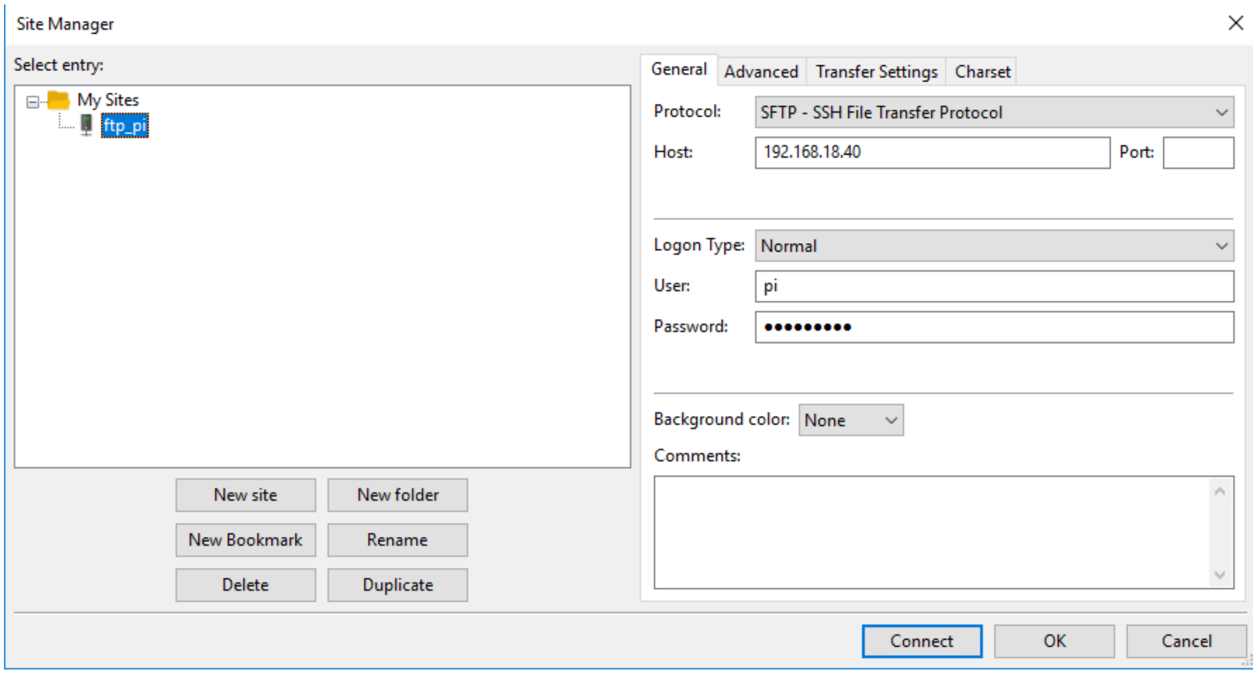
After a quick install open it up and now connect it to an FTP server. It has 3 ways to connect, here we use the **Quick Connect** bar. Enter the **hostname/IP**, **username**, **password** and **port (22)**, then click **Quick Connect** or press **Enter** to connect to the server.


sftp://pi@192.168.18.40 - FileZilla
File Edit View Transfer Server Bookmarks Help
Quickconnect

Host: sftp:// Username: pi Password: ••••••••• Port: 22

```
Status:     Connected to 192.168.18.40
Status:     Retrieving directory listing...
Status:     Listing directory /home/pi
Status:     Directory listing of "/home/pi" successful
```

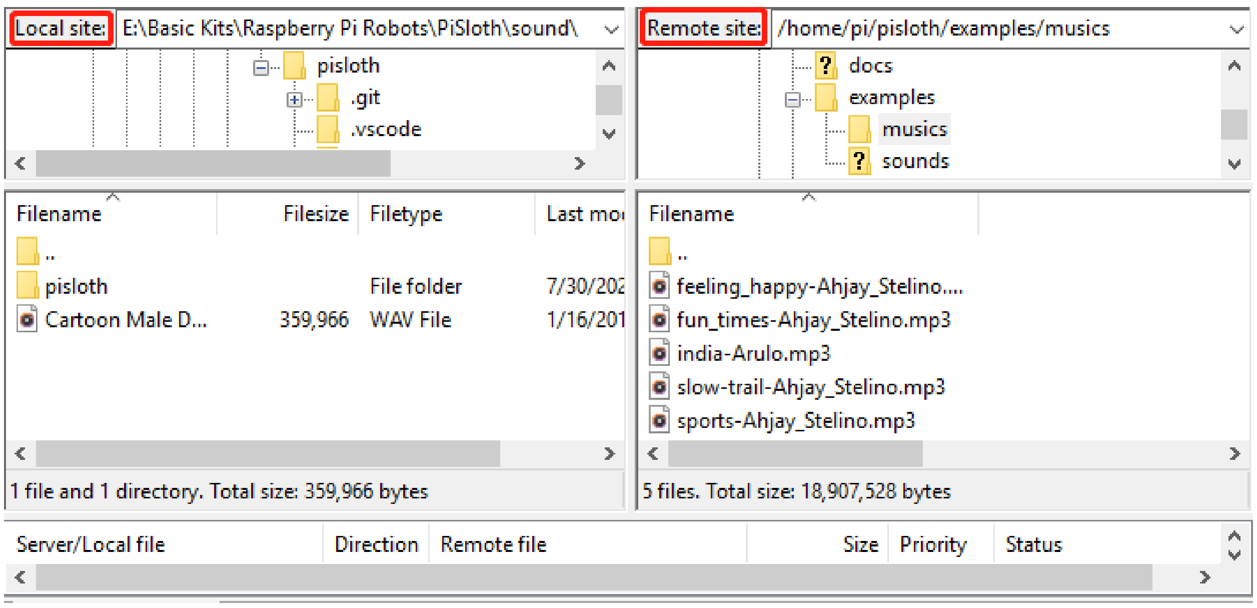
Note: Quick Connect is a good way to test your login information. If you want to create a permanent entry, you can select **File**-> **Copy Current Connection to Site Manager** after a successful Quick Connect, enter the name and click **OK**. Next time you will be able to connect by selecting the previously saved site inside **File** -> **Site Manager**.



Protocol: SFTP - SSH File Transfer Protocol
Host: 192.168.18.40 Port:
Logon Type: Normal
User: pi
Password: •••••••••
Background color: None

Step 3: Upload/download files.

You can upload local files to Raspberry Pi by dragging and dropping them, or download the files inside Raspberry Pi files locally.


Local site: E:\Basic Kits\Raspberry Pi Robots\PiSloth\sound\

pisloth
.git
.vscode

Remote site: /home/pi/pisloth/examples/musics

docs
examples
musics
sounds

Filename	Filesize	Filetype	Last mo
..			
pisloth		File folder	7/30/202
Cartoon Male D...	359,966	WAV File	1/16/201

1 file and 1 directory. Total size: 359,966 bytes

Filename
..
feeling_happy-Ahjay_Stelino....
fun_times-Ahjay_Stelino.mp3
india-Arulo.mp3
slow-trail-Ahjay_Stelino.mp3
sports-Ahjay_Stelino.mp3

5 files. Total size: 18,907,528 bytes

Server/Local file	Direction	Remote file	Size	Priority	Status

4.2 Get the IP address

There are many ways to know the IP address, and two of them are listed as follows.

Checking via the router

If you have permission to log in the router(such as a home network), you can check the addresses assigned to Raspberry Pi on the admin interface of router.

The default hostname of the Raspberry Pi OS is raspberrypi, and you need to find it. (If you are using ArchLinuxARM system, please find alarmpi.)

Network Segment Scanning

You can also use network scanning to look up the IP address of Raspberry Pi. You can apply the software, **Advanced IP scanner** and so on.

Scan the IP range set, and the name of all connected devices will be displayed. Similarly, the default hostname of the Raspberry Pi OS is raspberrypi, if you haven't modified it.

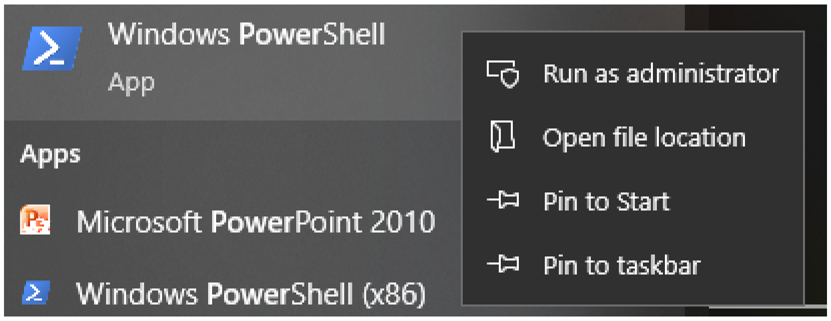
4.3 Install OpenSSH via Powershell

When you use `ssh <username>@<hostname>.local` (or `ssh <username>@<IP address>`) to connect to your Raspberry Pi, but the following error message appears.

```
ssh: The term 'ssh' is not recognized as the name of a cmdlet, function,␣
↪script file, or operable program. Check the
spelling of the name, or if a path was included, verify that the path is␣
↪correct and try again.
```

It means your computer system is too old and does not have OpenSSH pre-installed, you need to follow the tutorial below to install it manually.

1. Type `powershell` in the search box of your Windows desktop, right click on the `Windows PowerShell`, and select `Run as administrator` from the menu that appears.



2. Use the following command to install `OpenSSH.Client`.

```
Add-WindowsCapability -Online -Name OpenSSH.Client~~~~0.0.1.0
```

3. After installation, the following output will be returned.

```
Path          :
Online        : True
RestartNeeded : False
```

4. Verify the installation by using the following command.

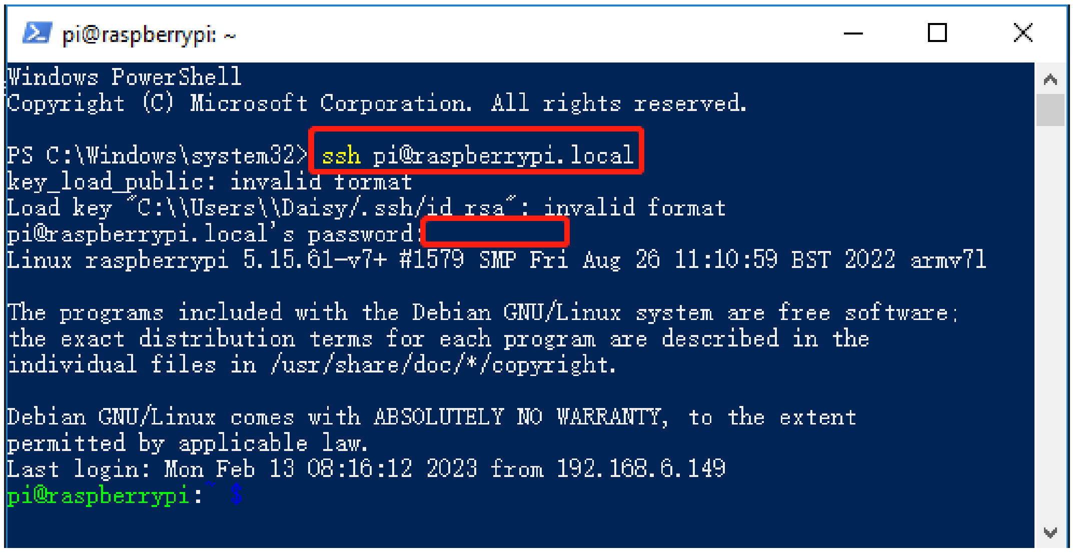
```
Get-WindowsCapability -Online | Where-Object Name -like 'OpenSSH*'
```

5. It now tells you that `OpenSSH.Client` has been successfully installed.

```
Name  : OpenSSH.Client~~~~0.0.1.0
State : Installed

Name  : OpenSSH.Server~~~~0.0.1.0
State : NotPresent
```

Warning: If the above prompt does not appear, it means that your Windows system is still too old, and you are advised to install a third-party SSH tool, like *PuTTY*.

6. Now restart PowerShell and continue to run it as administrator. At this point you will be able to log in to your Raspberry Pi using the `ssh` command, where you will be prompted to enter the password you set up earlier.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> ssh pi@raspberrypi.local
key_load_public: invalid format
Load key "C:\\Users\\Daisy/.ssh/id_rsa": invalid format
pi@raspberrypi.local's password:
Linux raspberrypi 5.15.61-v7+ #1579 SMP Fri Aug 26 11:10:59 BST 2022 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Feb 13 08:16:12 2023 from 192.168.6.149
pi@raspberrypi:~ $
```

4.4 PuTTY

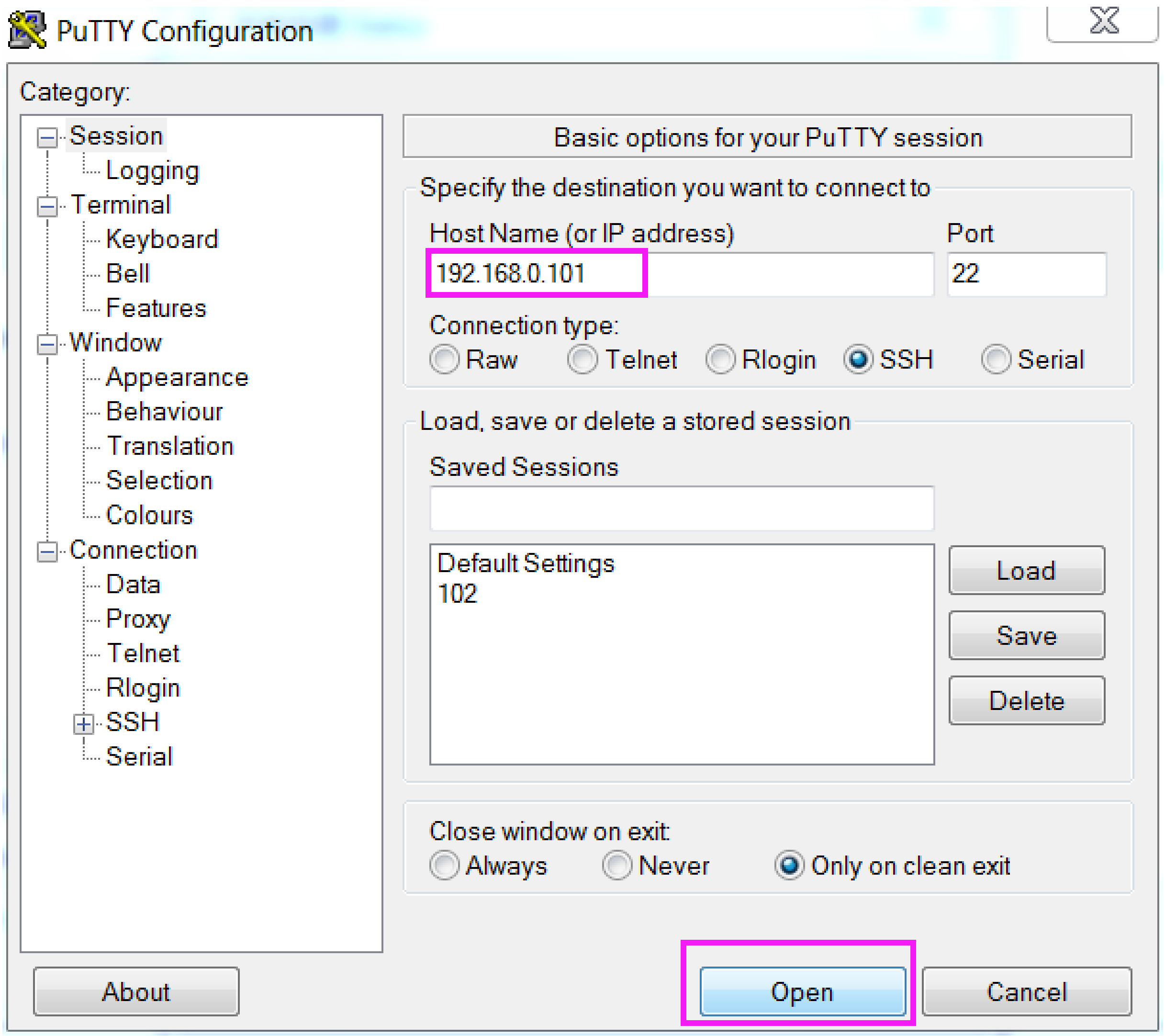
If you are a Windows user, you can use some applications of SSH. Here, we recommend PuTTY.

Step 1

Download PuTTY.

Step 2

Open PuTTY and click **Session** on the left tree-alike structure. Enter the IP address of the RPi in the text box under **Host Name (or IP address)** and **22** under **Port** (by default it is 22).



Specify the destination you want to connect to

Host Name (or IP address): 192.168.0.101

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions

Default Settings
102

Close window on exit: Always Never [✓ Only on clean exit]

Step 3

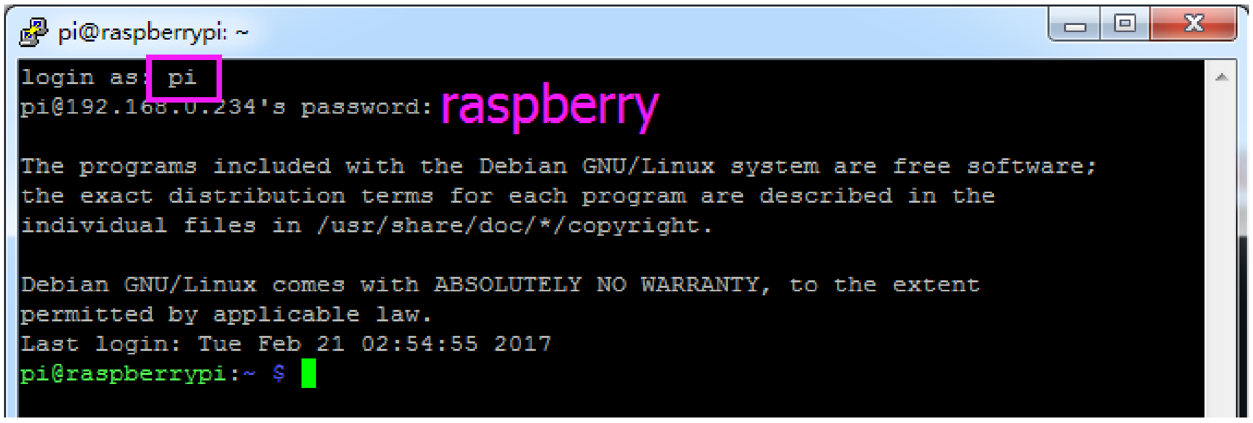
Click **Open**. Note that when you first log in to the Raspberry Pi with the IP address, there prompts a security reminder. Just click **Yes**.

Step 4

When the PuTTY window prompts "**login as:**", type in "**pi**" (the user name of the RPi), and **password**: "raspberry" (the default one, if you haven't changed it).

Note: When you input the password, the characters do not display on window accordingly, which is normal. What you need is to input the correct password.

If inactive appears next to PuTTY, it means that the connection has been broken and needs to be reconnected.



```
login as: pi
pi@192.168.0.234's password: raspberry

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Feb 21 02:54:55 2017
pi@raspberrypi:~ $
```

Step 5

Here, we get the Raspberry Pi connected and it is time to conduct the next steps.

CHAPTER

FIVE

COPYRIGHT NOTICE